E. U. V.
Estudis Universitaris de Vic
Escola Universitària Politècnica d'Osona
Adscrita a la Universitat Politècnica de Catalunya
Enginyeria Tècnica de Telecomunicació
especialitat en Sistemes de Telecomunicació

ADQUISICIÓ DEL SENYAL

Annex al manual de pràctiques

Antoni Suriñach i Albareda

ÍNDEX

Annex 1
L'AMPLIFICADOR D'INSTRUMENTACIÓ

1.- INTRODUCCIÓ.

Un amplificador d'instrumentació (A.I.) és un bloc que amplifica una tensió diferencial d'entrada per un guany precís i que presenta, alhora, les característiques següents:

- Una **impedància d'entrada molt elevada,** que assegura que el guany de l'amplificador no serà afectat per les impedàncies de la font de senyal (R_S). Aquesta impedància d'entrada elevada també assegura que el corrent de polarització d'entrada serà molt petit, minimitzant així les tensions d'offset d'entrada originades per ($I_B \cdot R_S$).

- Una **impedància de sortida molt baixa** que assegura que la tensió de sortida no serà afectada per la impedància de càrrega.

- Una **forta realimentació negativa,** la qual proporciona una linealitat excel.lent en el guany, tot i que aquest pugui ser elevat. El guany de l'amplificador és variable i se sol ajustar amb un o dos resistors exteriors.

- Un **gran aparellament dels diferents components interns i una petita tolerància dels mateixos,** de manera que les tensions i els corrents d'offset són molt baixos, les derives petites, i la relació de refús en mode comú molt elevada.

Degut a aquestes característiques, si les dues tensions d'entrada són iguals:

$$V^+ = V^- = V_{mc} \quad \text{(tensió en mode comú)} \qquad (E1)$$

la sortida de l'amplificador d'instrumentació serà igual a zero. El guany diferencial es fixa per resistències externes o mitjançant ponts, i sol variar entre 0,1 V/V i 1000 V/V.

En el mercat es disposa d'amplificadors d'instrumentació híbrids i monolítics. De tota manera, els A.I. són fàcils de realitzar amb components discrets i s'aconsegueixen bons resultats.

Els A.I. tenen un terminal de mostreig (SENSE) i un terminal de referència (REFERENCE). Com s'indica en la figura 1, els terminals SENSE i OUTPUT es connecten generalment entre ells, així com els terminals REFERENCE i TERRA, i la càrrega es connecta entre les dues unions anteriors.

L'amplificador satisfà:

$$V_{OUT} = G_d \cdot V_{IN} \qquad (E2)$$

i es defineixen la tensió diferencial d'entrada i la tensió en mode comú com:

$$V_d = V_{IN}^+ - V_{IN}^- \quad \text{(tensió diferencial d'entrada)}$$

$$V_{mc} = \frac{V_{IN}^+ + V_{IN}^-}{2} \quad \text{(tensió en mode comú)} \qquad (E3)$$

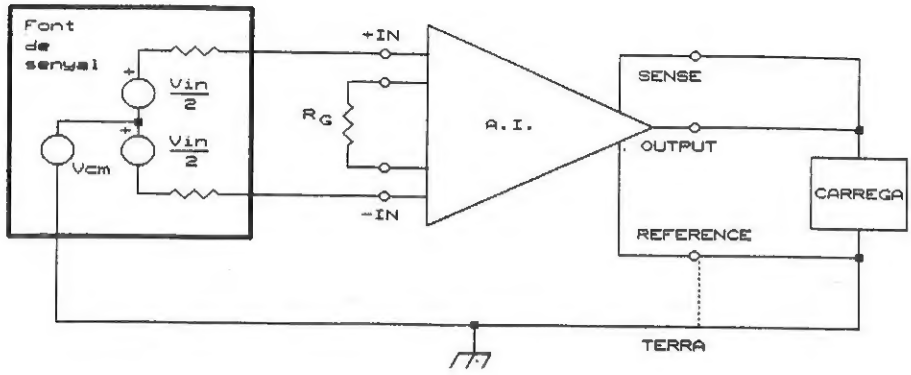


Figura 1. Amplificador d'Instrumentació.

Els terminals SENSE i REFERENCE són útils per manejar càrregues llunyanes amb elevats corrents. Aquestes càrregues es connecten a l'amplificador d'instrumentació amb quatre fils, de manera que els dos cables connectats a OUTPUT i TERRA són els que transporten el corrent, mentre que els connectats a SENSE i REFERENCE són els que sensen la tensió entre extrems de la càrrega (sense que es produeixi caiguda de tensió en ells) i proporcionen la realimentació adequada a l'A.I. per compensar la caiguda de tensió en els cables OUTPUT i TERRA deguda al pas del corrent, com es veu en la figura 2.

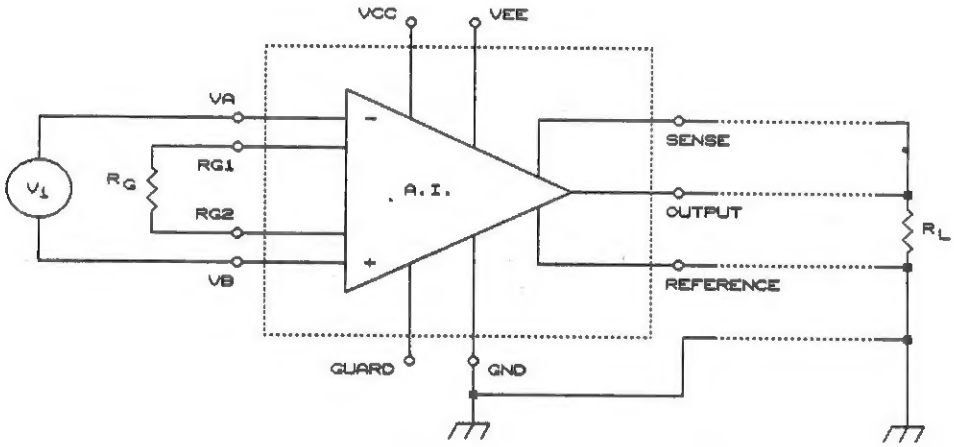


Figura 2. Connexió d'una càrrega llunyana a l'A.I.

Les dues figures següents mostren dues aplicacions comunes dels terminals SENSE i REFERENCE:

a) Amplificador d'instrumentació amb un amplificador de corrent a la sortida (figura 3). En aquest circuit s'observa que el senyal de sortida de l'A.I s'amplifica en intensitat mitjançant un amplificador de corrent X1, mantenint, però, la mateixa tensió final mercès a la realimentació de SENSE.

b) Amplificador d'instrumentació amb offset de sortida variable (figura 4). En aquest circuit s'afegeix un offset a la tensió de sortida aplicant aquesta tensió d'offset al terminal REFERENCE mitjançant un seguidor de tensió.

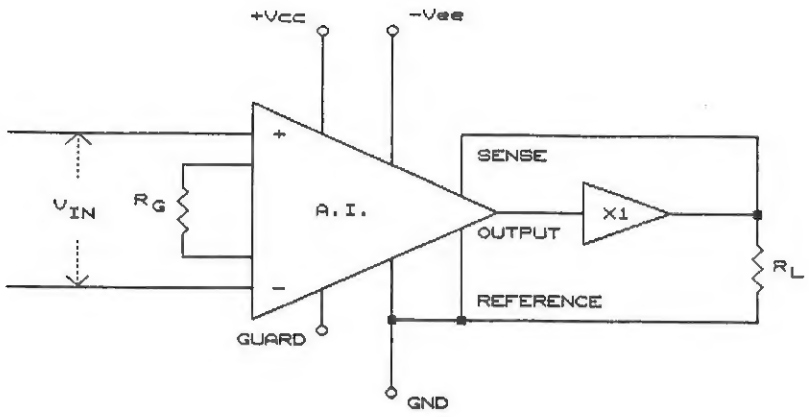


Figura 3. Amplificador d'instrumentació amb amplificador de corrent.

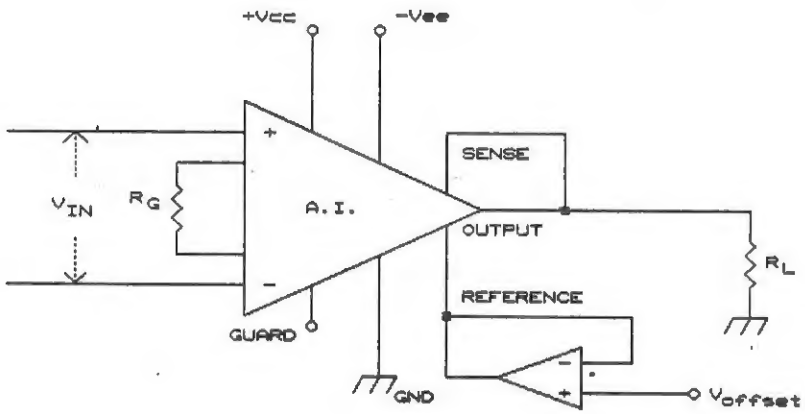


Figura 4. Amplificador d'instrumentació amb offset de sortida variable.

El **GUANY EN MODE COMÚ (G_{mc})** es defineix com la relació entre la tensió de sortida i la tensió en mode comú de l'entrada, quan la tensió diferencial a l'entrada és zero.

$$G_{mc} = \frac{V_O}{V_{mc}}\bigg|_{V_d=0} \qquad (E4)$$

El **GUANY DIFERENCIAL (G_d)** es defineix com la relació entre la tensió de sortida i la tensió diferencial de l'entrada, quan la tensió en mode comú a l'entrada és zero.

$$G_d = \frac{V_O}{V_d}\bigg|_{V_{mc}=0} \qquad (E5)$$

La **RELACIÓ DE REFÚS EN MODE COMÚ (CMRR)** es defineix com la relació entre el guany diferencial i el guany en mode comú, i es sol expressar en decibels:

$$CMRR \quad 20 \log \frac{G_d}{G_{mc}} \quad (dB) \qquad (E6)$$

El refús en mode comú indica la capacitat de l'amplificador de refusar les tensions d'entrada en mode comú (les que són presents, alhora, en les dues entrades).

Les característiques esmentades d'un amplificador d'instrumentació el fan molt útil per amplificar senyals de sortida de transductors de baix nivell, com ara termoparells, ponts de galgues extensomètriques, terminals biològics, etc, que produeixen senyals diferencials molt petits superposats a tensions en mode comú de polarització. A més, el soroll de massa en mode comú és freqüent en moltes aplicacions.

Per tal de minimitzar l'efecte de sorolls i interferències és important seguir els següents consells:

- Utilitzar el terminal de GUARD sempre que sigui possible.

- Si és possible, s'ha d'alimentar la part analògica i la part digital d'un circuit amb dues fonts d'alimentació diferents (els circuits digitals generen molt soroll).

- Els terres dels diferents components han d'anar connectats a un sol punt, que serà el terra de la font d'alimentació (si hi ha més d'una font, els terres dels components alimentats per cada font aniran connectats només al terra de la seva font). Així mateix, si hi ha vàries fonts d'alimentació, els seus respectius terres aniran connectats a un sol punt que serà el terra de l'equip.

- Tots els C.I. es desacoblaran amb condensadors petits (per exemple condensadors ceràmics de 47 nF) per filtrar el soroll d'alta freqüència. Així mateix, es disposaran condensadors elevats en els busos d'alimentació per filtrar les baixes freqüències (per exemple, condensadors de tàntal de 4,7 o 10 μF).

- És aconsellable separar la part digital de l'analògica en la placa de circuit imprès, així com separar els blocs per velocitat (els circuits més ràpids seran els més propers als connectors d'entrada/sortida i els més lents seran els més allunyats).

- La distribució de l'alimentació per la placa del circuit imprès es farà mitjançant busos paral.lels (formant una línia de transmissió de petita impedància), a fi de minimitzar la diafonia.

En la figura 5 es pot veure el connexionat de terres en un sistema d'adquisició de dades.

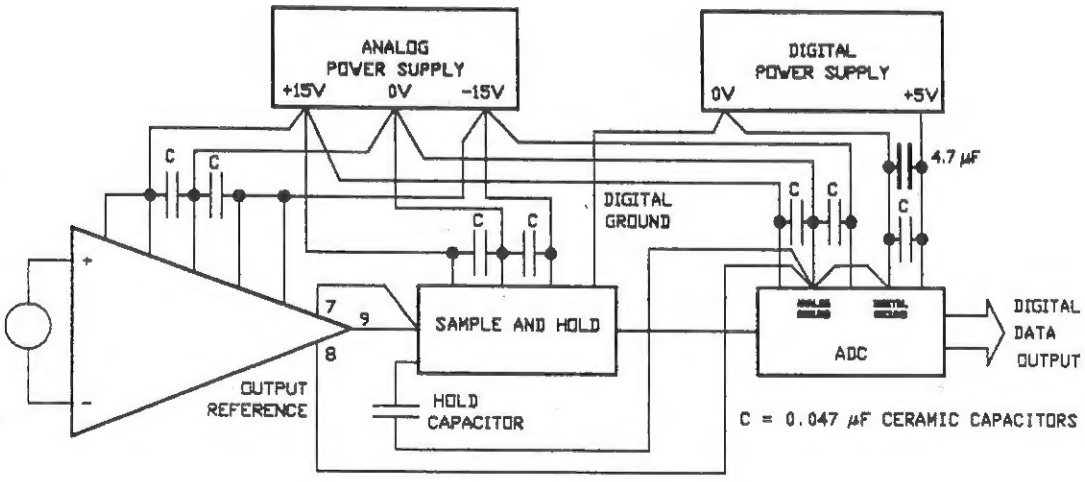


Figura 5. Connexions de terres en un sistema d'adquisició de dades.

2.- AMPLIFICADOR D'INSTRUMENTACIÓ REALITZAT AMB COMPONENTS DISCRETS.

Amb tres amplificadors operacionals es pot implementar un amplificador d'instrumentació de bones característiques. En la figura 6 es pot veure l'esquema d'un A.I. realitzat amb tres Op-Amp (el quart amplificador operacional s'utilitza per obtenir el senyal de GUARD actiu), amb les següents característiques elèctriques:

- Entrada diferencial de senyal.

- Sortida referida a massa.

- Guany en tensió variable mitjançant l'ajust d'un resistor exterior.

- Ajust de les tensions d'offset dels operacionals.

- Refús del mode comú elevat, amb la possibilitat d'ajustar-lo.

- Gran impedància d'entrada, ja que els senyals s'introdueixen a través dels seguidors de tensió A1 i A2.

- Baixa impedància de sortida.

- Terminals SENSE i REFERENCE accessibles, amb la possibilitat d'introduir tensions d'offset a la sortida i de compensar caigudes en càrregues llunyanes.

- Terminal de GUARD actiu accessible, amb la tensió en mode comú. Aquesta característica permet eliminar gran part dels sorolls i interferències en mode comú que es produeixen quan es tracta amb senyals dèbils i llunyans.

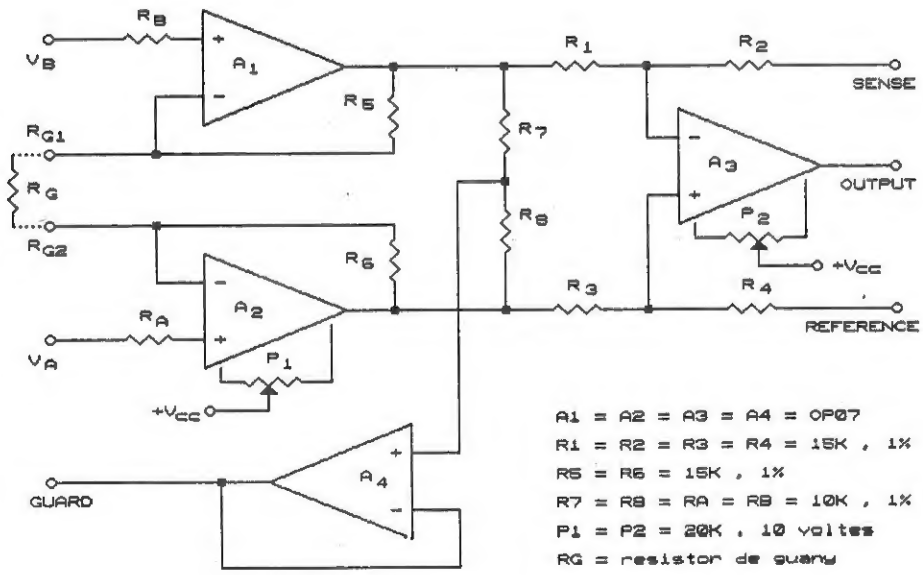


Figura 6. Amplificador d'instrumentació construït amb components discrets.

Si considerem que:

$$R_1 = R_3$$
$$R_2 = R_4$$
$$R_5 = R_6 \qquad (E7)$$

la tensió de sortida (V_{OUT}) que s'obté connectant SENSE amb OUTPUT i REFERENCE amb MASSA, i disposant un resistor exterior R_G ve donada per l'expressió:

$$V_{OUT} = \frac{R_2}{R_1}\left[\frac{2R_5}{R_G} + 1\right](V_B - V_A) \qquad (E8)$$

I escollint:

$$R_7 = R_8 \qquad (E9)$$

la tensió en el terminal GUARD és:

$$V_{GUARD} = \frac{V_B + V_A}{2} \qquad (E10)$$

Aquest circuit el podem modelar de la forma indicada en la figura 7.

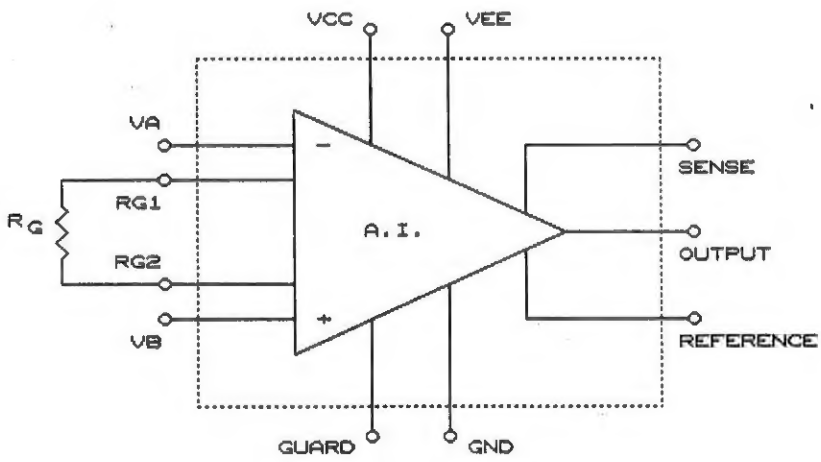


Figura 7. Amplificador d'instrumentació modelat.

2.1.- AJUST DE L'OFFSET.

Aquest ajust s'aconsegueix quan connectant ambdós terminals d'entrada a massa, la tensió de sortida és nul.la. S'ha d'ajustar P_1 i P_2 fins que s'aconsegueixi.

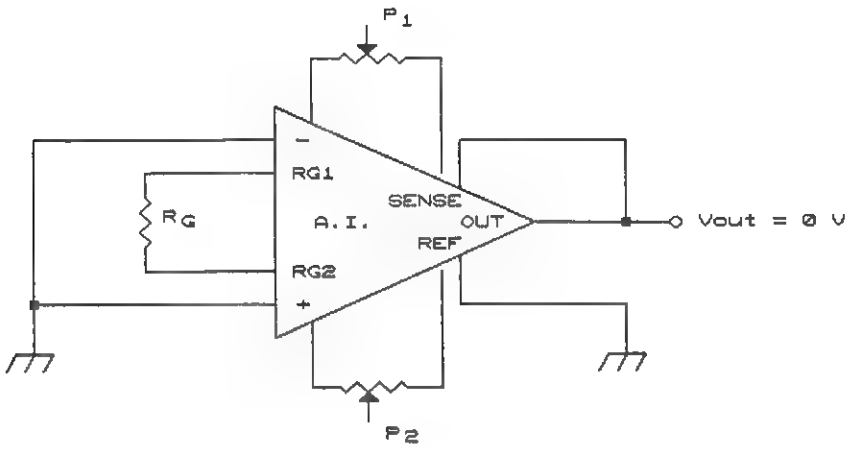


Figura 8. Circuit per ajustar l'offset.

2.2.- AJUST DEL GUANY DIFERENCIAL.

Aquest ajust s'aconsegueix calculant R_G en l'equació (E8) per tal d'aconseguir el valor d'amplificació diferencial que requereix la nostra aplicació. En la pràctica es pot utilitzar per R_G un potenciòmetre ajustable multivolta, i muntar el circuit de la figura 9. Desplaçant el cursor del potenciòmetre P, es mesura una determinada tensió diferencial V_d, i es varia R_G fins obtenir el guany desitjat.

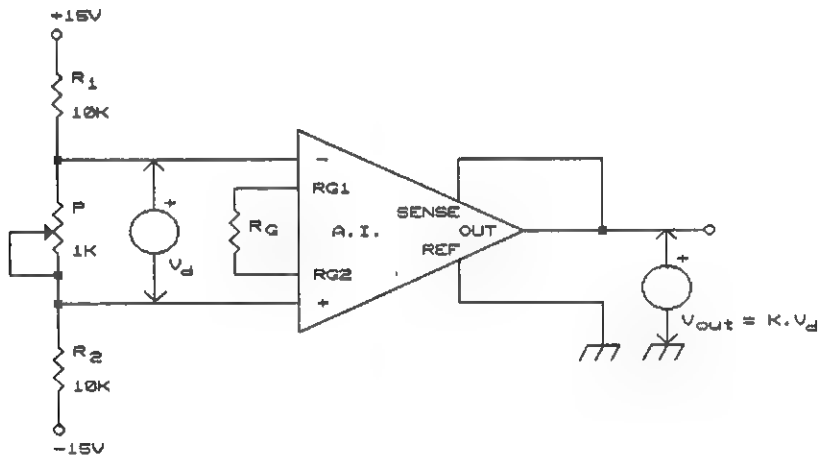


Figura 9. Circuit per ajustar el guany diferencial.

2.3.- AJUST DEL GUANY EN MODE COMÚ.

És convenient que el guany en mode comú sigui el mínim possible, de tal manera que el CMRR sigui màxim. Això s'aconsegueix quan es satisfà l'equació:

$$\frac{R_2}{R_1} = \frac{R_4}{R_3} \qquad (E11)$$

Com que és impossible que els resistors estiguin totalment aparellats, es sol col.locar una petita resistència ajustable entre REFERENCE i MASSA, o be entre SENSE i OUTPUT, de manera que es satisfaci:

$$\frac{R_2}{R_1} = \frac{(R_4 + P)}{R_3} \qquad (E12)$$

La millor manera d'ajustar el guany en mode comú consisteix en col.locar una tensió en mode comú equivalent a la que existirà en el circuit quan estigui funcionant. A partir d'aquí s'actua sobre el resistor ajustable connectat en sèrie amb R4 fins obtenir una tensió de sortida de 0V. En el cas que no sigui possible obtenir aquest valor, el resistor ajustable es retirarà de la branca d'R4 i es connectarà en la branca d'R2.

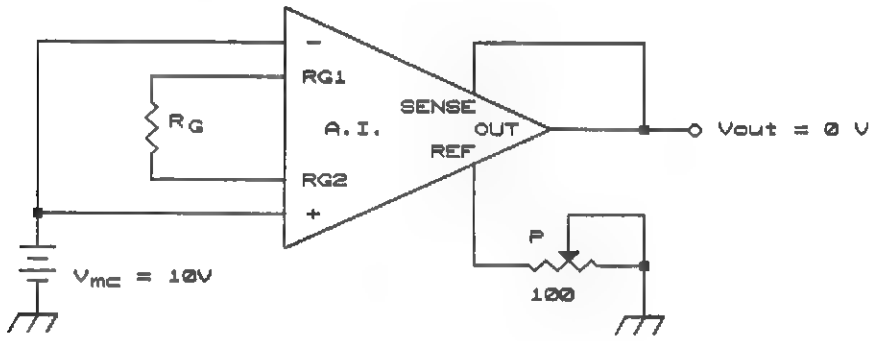


Figura 10. Circuit per ajustar el guany en mode comú.

Annex 2
MINIMITZACIÓ DE ΔR_{ON} EN ELS COMMUTADORS ANALÒGICS CMOS

1.- INTRODUCCIÓ.

Els commutadors analògics CMOS són àmpliament utilitzats per a realitzar circuits amb aplicacions de multiplexació i funcions de commutació. Idealment tenen resistència nul.la quan estan tancats, resistència infinita quan estan oberts, no tenen fuites, són de resposta instantània i no tenen capacitats paràsites. Encara que aquestes suposicions són raonablement vàlides en aplicacions de baixa freqüència, a partir de mitjanes freqüències el bon dissenyador sempre les posarà en dubte per determinar quina quantitat d'error s'introdueix i, fins i tot, quina configuració és la més adequada.

2.- CIRCUIT DE COMMUTACIÓ.

L'esquema de la Figura 1 és una aproximació raonable d'un circuit commutador CMOS, aïllat dielèctricament en un únic estrat. L'aïllament dielèctric permet protegir el circuit dels pics de commutació i de sobretensions (fins $\pm 25V$) provinents de la font d'alimentació. Cal tenir en compte que la conducció es realitzarà pel FET de canal N, o pel FET de canal P, segons quina sigui la polaritat del senyal a commutar. Aquest detall cal tenir-lo present, ja que els dos FET no són perfectament simètrics.

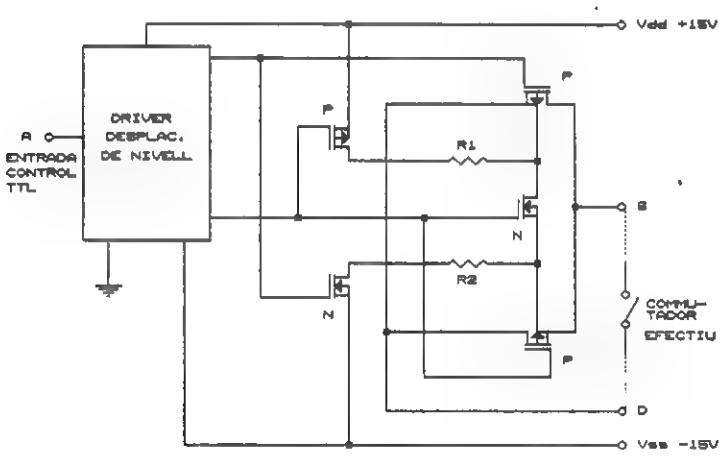
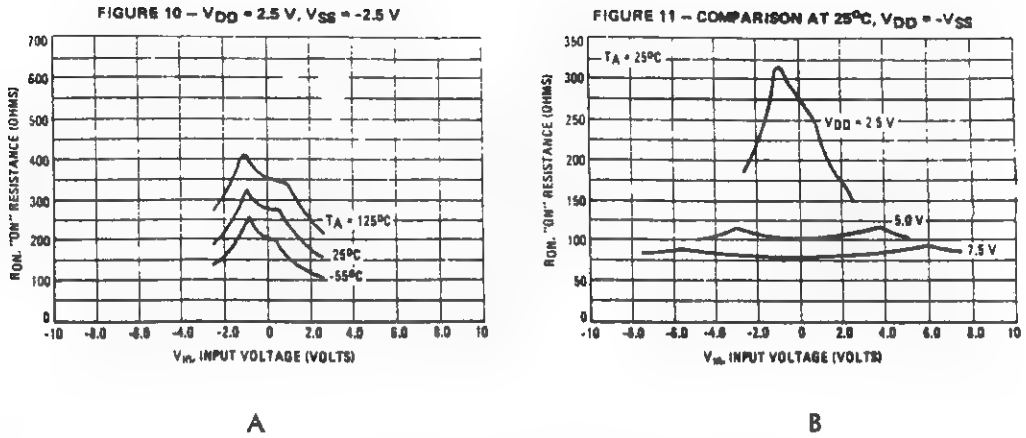


Figura 1. Circuiteria de sortida típica d'un commutador CMOS.

La Figura 2 presenta les corbes típiques de la resistència del commutador en estat de conducció (Ron), tal com apareixen en els fulls de característiques del producte. Es pot veure com Ron queda afectada per la tensió del senyal d'entrada, per la tensió d'alimentació i per la temperatura. Quanta més tensió d'alimentació i menys temperatura hi hagi, més baixa i menys depenent del senyal serà Ron.



A B

Figura 2. A - Ron respecte de Vin, en funció de +Vdd (-Vss).
B - Ron respecte de Vin, en funció de la temperatura.

De quina manera es pot minimitzar la influència de les variacions d' Ron?

En la Figura 3 hi ha un circuit amplificador inversor amb quatre entrades commutades. La Ron, que està en sèrie amb la resistència d'entrada de 10 KΩ, afecta el guany del circuit. Fins i tot, malgrat que s'hagi compensat per una tensió d'alimentació i una entrada analògica determinades, qualsevol variació en el senyal d'entrada provocarà variacions en el guany de l'amplificador i disminuirà la seva precisió.

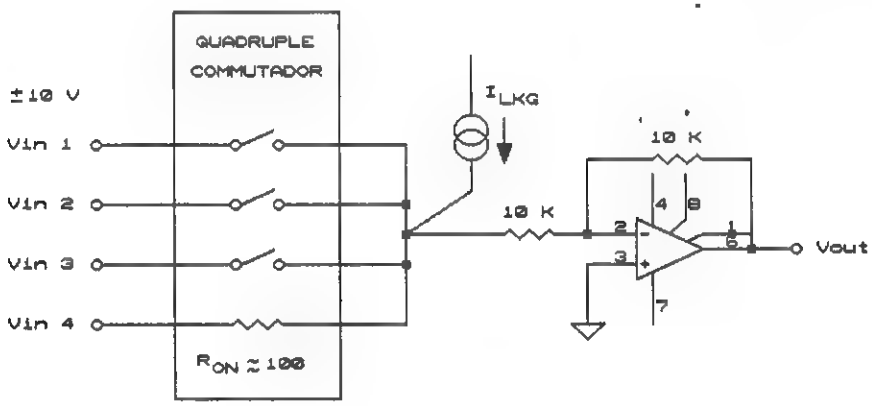


Figura 3. Inversor de guany unitari amb commutadors d'entrada.

La solució més senzilla, si l'amplificador no ha d'invertir o ha d'actuar com un esmorteïdor de precisió, és utilitzar la configuració de no inversor, tal com es veu en la Figura 4. Com que no hi ha resistència en sèrie, el guany no queda afectat.

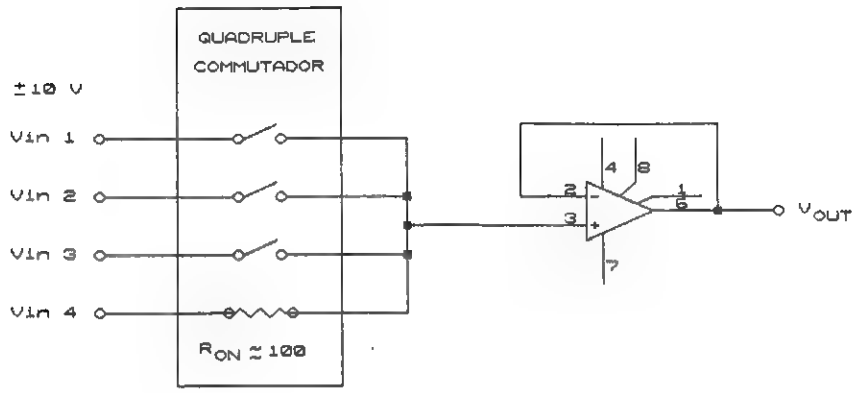


Figura 4. Solució amb configuració no inversora.

En el cas d'utilitzar l'amplificador sumador, la solució consisteix en connectar els commutadors al punt sumatori de l'amplificador, deixant les resistències d'entrada abans dels commutadors (veure la Figura 5). D'aquesta manera les variacions dels senyals d'entrada queden reduïdes, en el commutador, a variacions de magnitud tres o quatre ordres inferiors, minimitzant així les variacions de Ron per culpa, precisament, d'aquestes variacions dels senyals d'entrada.

Malauradament, aquesta solució pot perjudicar l'ampiada de banda de l'amplificador, ja que la capacitat Cs pot requerir, per compensar, una capacitat en paral.lel amb la resistència de realimentació.

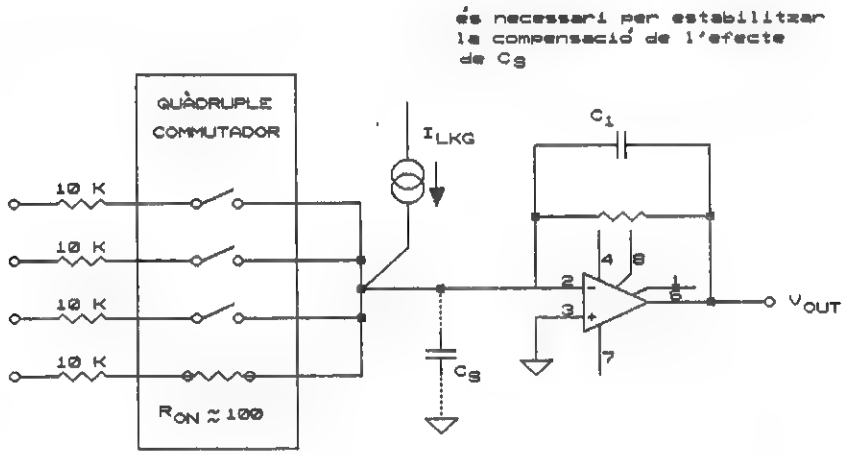


Figura 5. Connexió dels commutadors al punt sumatori.

Una altra possible solució és utilitzar valors molt elevats en les resistències d'entrada i de realimentació (Figura 6). Llavors les variacions de Ron seran negligibles comparades amb el valor de les altres resistències. Aquesta solució només té l'inconvenient que l'amplada de banda quedarà afectada per una gran constant de temps RC.

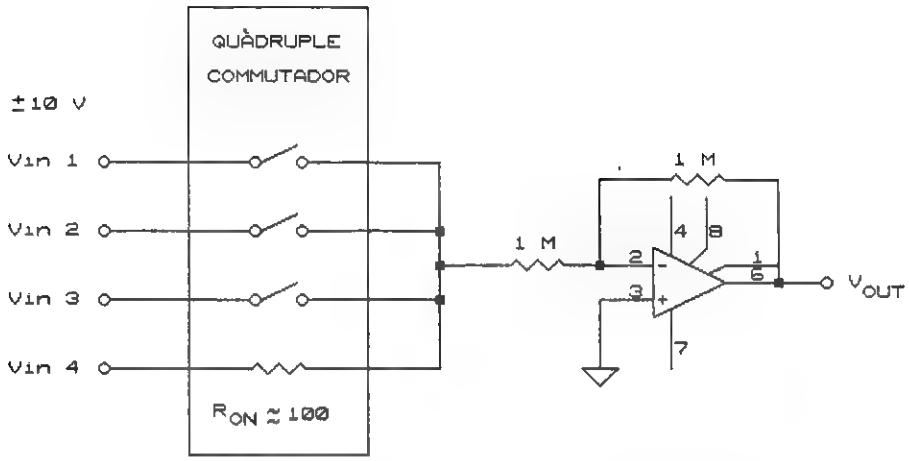


Figura 6. Utilització de valors molt elevats de resistència.

Els circuits de les figures 5 i 6 no compensen els efectes de la variació de Ron respecte de la temperatura. Un circuit que té una bona compensació és el que està representat en la Figura 7. En aquest circuit s'utilitza un dels commutadors, posat sempre en estat de conducció (ON), en sèrie amb la resistència de realimentació. Aleshores la Ron d'aquest commutador tindrà les mateixes variacions, per efecte de la temperatura, que les de la resta de commutadors. D'aquesta manera, les resistències d'entrada i de realimentació s'autocompensaran i mantindran el guany constant.

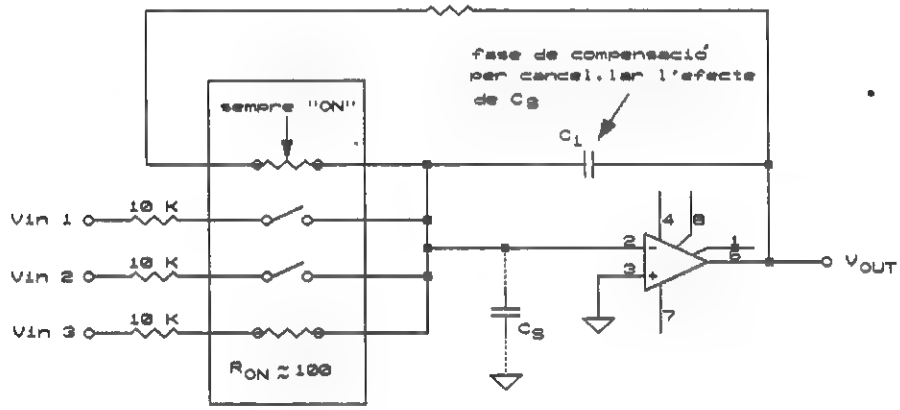


Figura 7. Commutadors en sèrie, per compensar el guany.

El principal efecte en c.c., en l'estat de no conducció (OFF), és que el corrent de fuites polaritzarà la sortida del circuit. La polaritat de l'error vindrà determinada per la polaritat del corrent de fuita més gran.

Annex 3
TRANSMISSIÓ DEL SENYAL

Un problema freqüent en sistemes d'instrumentació i control apareix al fer la mesura de senyals analògics provinents de transductors de baix nivell de sortida i distants dels circuits de procés (distàncies superiors als 4 metres). Aquest problema s'agreuja quan els senyals analògics han de transitar per llocs on hi ha sorolls elèctrics d'elevat nivell, tals com els que es troben en sistemes industrials amb maquinària elèctrica pesant.

En els propers apartats veurem dos dels mètodes més freqüents utilitzats en la transmissió dels senyals: la transmissió en llaç de corrent, i la transmissió en freqüència.

1.- TRANSMISSIÓ EN LLAÇ DE CORRENT.

La transmissió del senyal en llaç de corrent és una solució al problema plantejat. L'estàndard més usat és el de 4-20 mA, amb un marge o span de 16 mA i un offset de 4 mA.

La transmissió per corrent proporciona un alt nivell d'immunitat al soroll a causa que la informació rebuda no es veu afectada per les caigudes de tensió en els cables de connexió, ni pels termoparells paràsits, ni per les resistències dels contactes ni les tensions de soroll induïdes. Al mateix temps, l'offset proporciona una distinció entre el zero (representat per 4 mA) i la no-informació motivada pel circuit obert (no circula corrent).

Un altre avantatge d'aquesta forma de transmissió és que per algunes aplicacions es pot subministrar alimentació al transductor a través dels 4 mA de corrent que no es fan servir en la transferència d'informació. Per tant, la informació es transmet en un sentit i l'alimentació en el sentit contrari, i només calen dos conductors per la transmissió.

Finalment, la informació en forma de corrent permet que diverses càrregues situades en punts diferents puguin connectar-se en sèrie fins a la tensió màxima especificada. Per exemple, la sortida d'un transductor pot enviar informació a un enregistrador i a un mesurador, i proporcionar la informació d'entrada d'un controlador.

La figura 1 mostra una aplicació típica d'un llaç de 4-20 mA emprant un mòdul convertidor tensió-corrent.

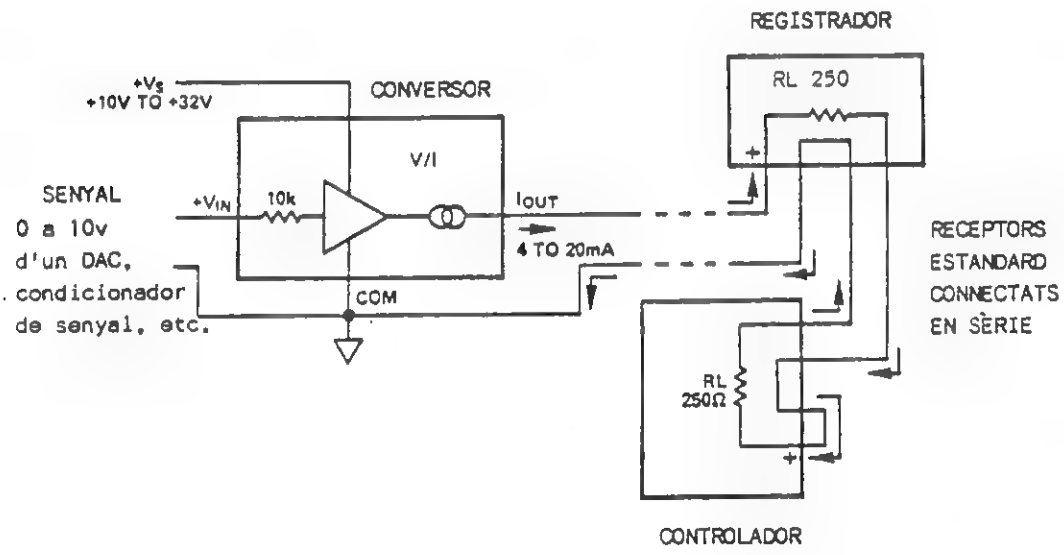


Figura 1. Llaç de corrent 4-20 mA.

La figura 2 correspon a un circuit de translació d'un senyal d'entrada de 0 a 10 V a un senyal de sortida de 4-20 mA.

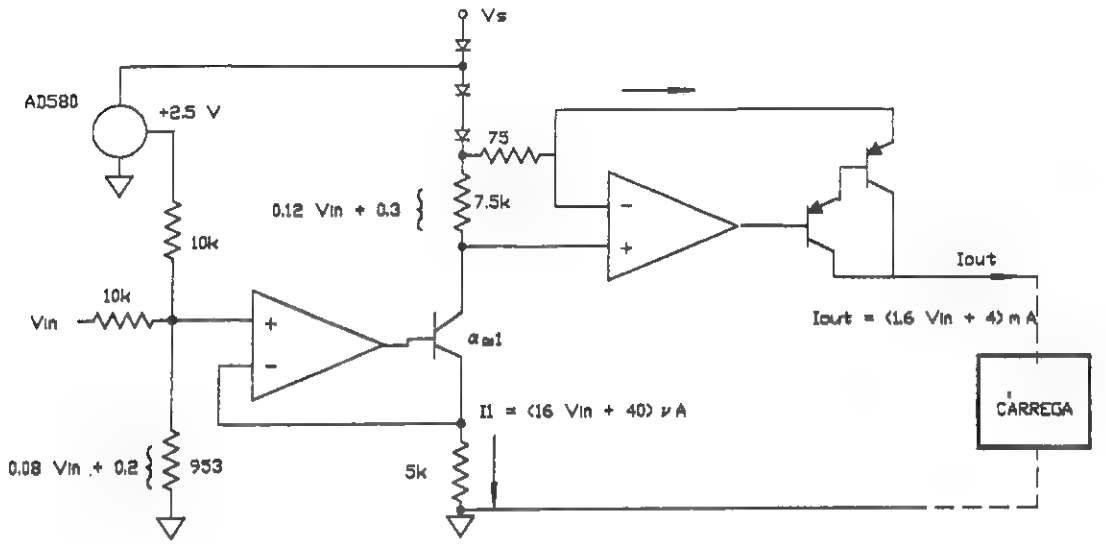


Figura 2. Circuit de translació 0-10 V a 4-20 mA.

La figura 3 mostra un circuit transmissor de corrent 4-20 mA que fa servir l'offset per alimentar el circuit transmissor.

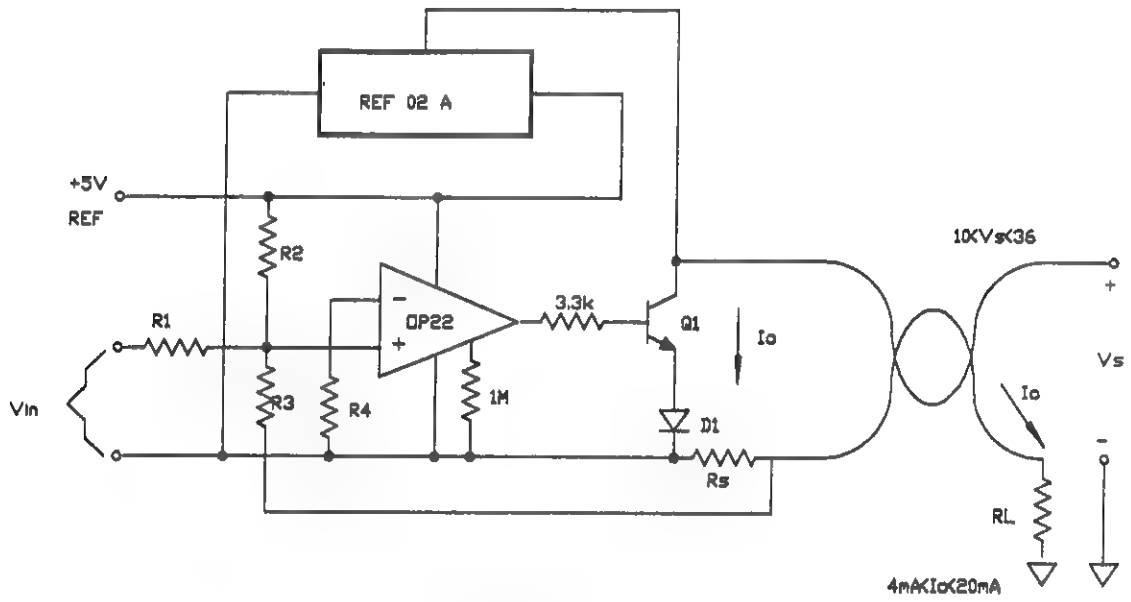


Figura 3. Transmissió 4-20 mA amb dos fils.

El llaç de corrent pot alimentar-se amb una tensió no regulada de 10 a 36 volts de contínua. Per realitzar "l'alimentació flotant" cal fer servir un amplificador operacional programable (tipus OP22) i un circuit de referència de tensió (REF 02). El transmissor només fa servir 2 mA, i pot, per tant, subministrar els 2 mA restants (a 5 V) al transductor o circuit en pont sense excedir el corrent mínim del llaç de 4 mA. L'amplificador operacional regula el corrent de sortida I_O, per tal de satisfer la suma de corrents en mode no inversor.

$$\frac{V_{IN}}{R1} + \frac{5V}{R2} - \frac{I_O R_S}{R3} = 0 \qquad (E13)$$

$$I_O = \frac{1}{R_S} * \left(\frac{R3}{R1} V_{IN} + \frac{R3}{R1} 5V \right) \qquad (E14)$$

Aplicant aquestes equacions a un corrent de sortida de 4-20 mA i un senyal d'entrada de 0 a 100 mV, obtenim les relacions:

$$\frac{R3}{R1} = 16 \qquad \frac{R3}{R2} = 0{,}08 \qquad (E15)$$

Per minimitzar els efectes dels corrents de polarització d'entrada cal fer R4 igual al paral.lel de les resistències R1, R2 i R3.

2.- TRANSMISSIÓ DEL SENYAL EN FREQÜÈNCIA.

La transmissió del senyal en freqüència presenta una alta immunitat al soroll i permet transmetre la informació a distàncies considerables. Un convertidor tensió-freqüència converteix tensions analògiques o nivells de corrent en una sèrie d'impulsos compatibles TTL a una freqüència directament proporcional al senyal analògic. La sortida segueix contínuament el senyal d'entrada i respon directament als canvis d'aquesta, sense necessitat d'un rellotge de sincronització.

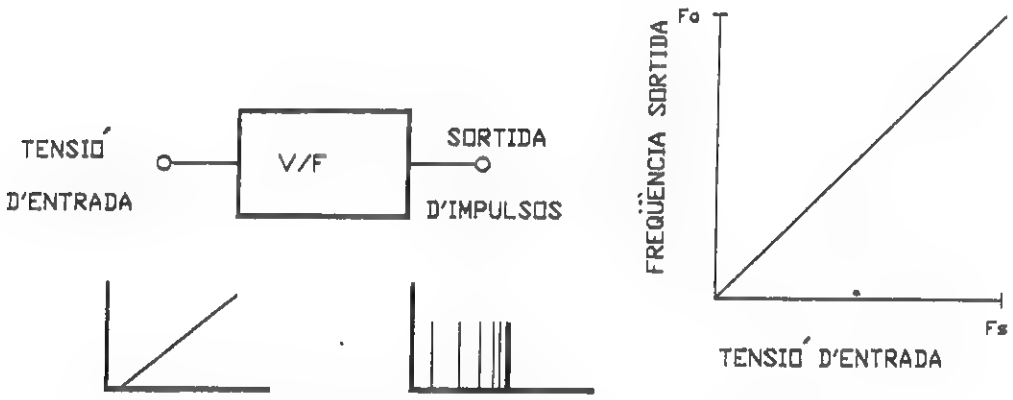


Figura 4. Conversió V/F ideal. Funció de transferència.

L'única restricció en aquesta forma de transmissió és aquella en la qual el senyal d'entrada canvia tan ràpidament que fa impossible que el convertidor el segueixi.

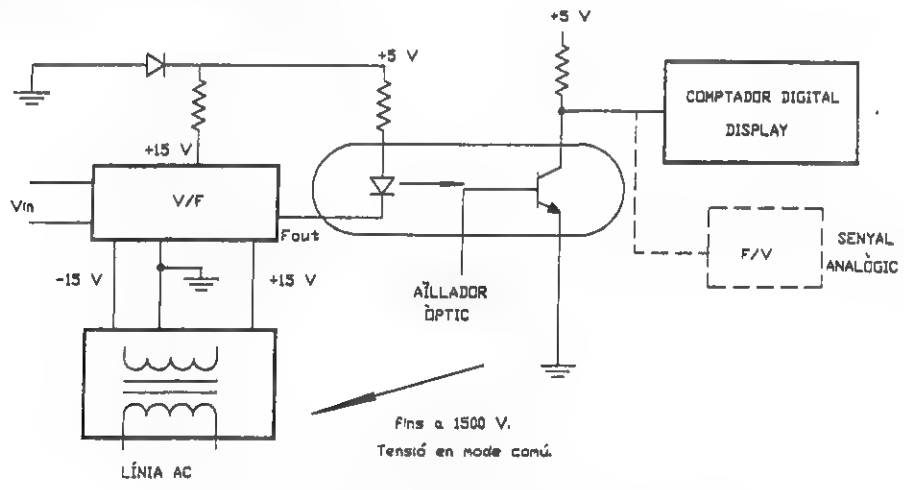


Figura 5. Aïllament galvànic. Transmissió per freqüència.

Un avantatge addicional en la transmissió per freqüència és la possibilitat d'aïllar galvànicament el circuit de procés del transductor per mitjà de la utilització d'un optoaïllador, tal com es representa en la figura 5.

Si el senyal analògic ha d'aparèixer a través d'un display numèric, o bé s'ha de processar amb un microprocessador, els impulsos es compten dins d'un període de temps fix (base de temps) mitjançant un comptador digital. La sortida del comptador pot aplicar-se a un display o bé a un port d'entrada del microprocessador (figura 6).

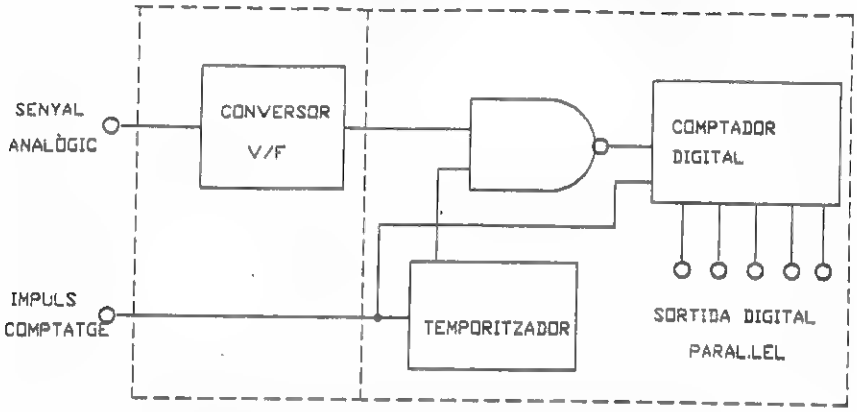


Figura 6. Convertidor V/F utilitzat com a convertidor A/D.

Si el senyal transmès en freqüència s'ha de convertir de nou a analògic, hem d'emprar un circuit convertidor freqüència-tensió amb una funció de transferència inversa a la del convertidor V/F (figura 7).

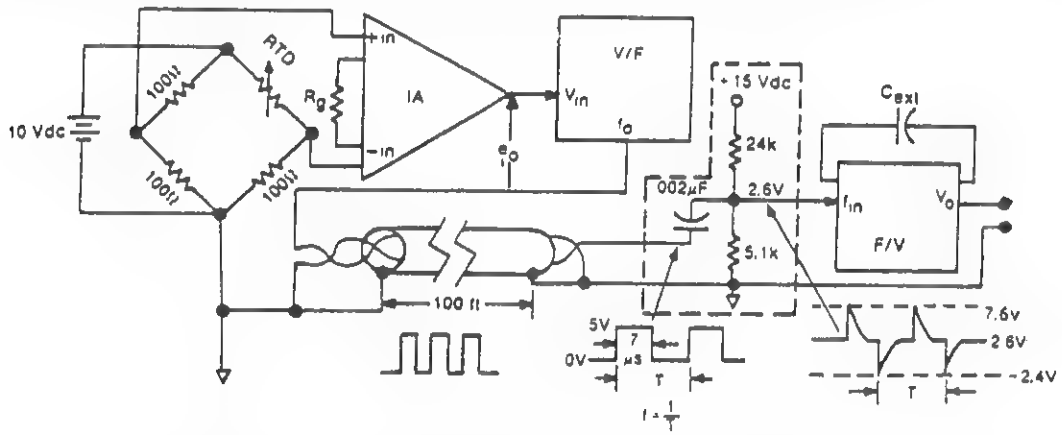


Figura 7. Transmissió en freqüència amb conversió final F/V.

Si es vol realitzar una transmissió d'informació amb elevada immunitat al soroll es poden utilitzar les dues tècniques, de llaç de corrent i de freqüència, tal com es veu en la figura 8.

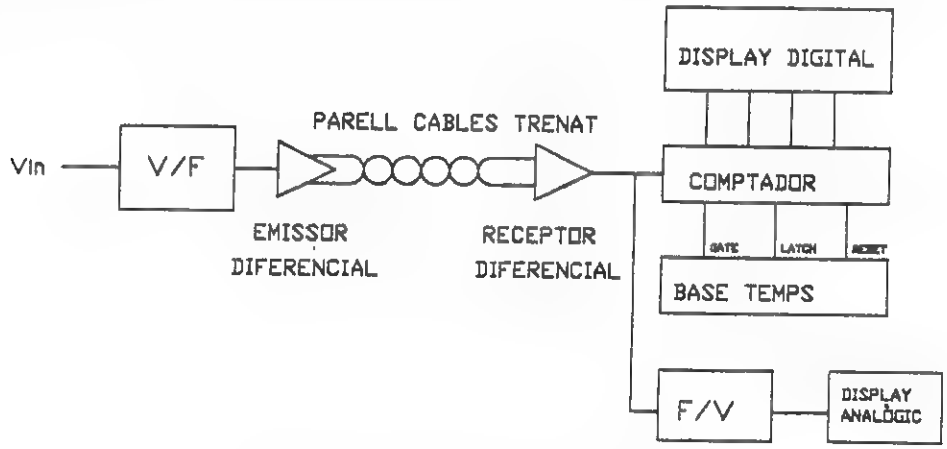
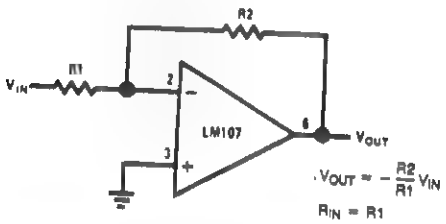


Figura 8. Transmissió per freqüència i corrent.

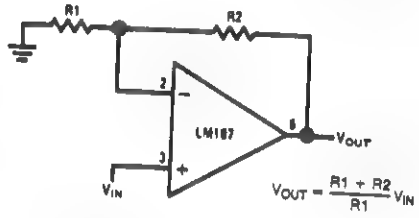
National Semiconductor
Application Note 31

SECTION 1—BASIC CIRCUITS

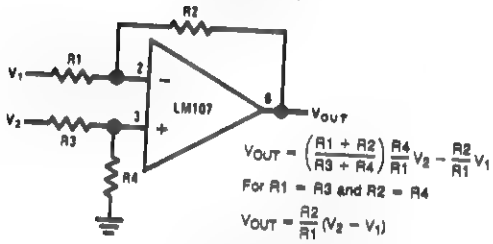
Inverting Amplifier



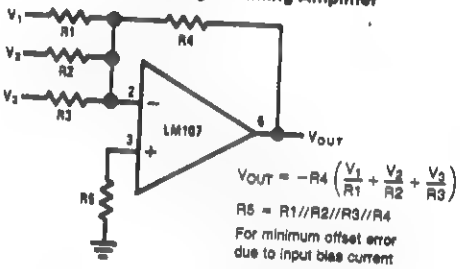
Non-Inverting Amplifier



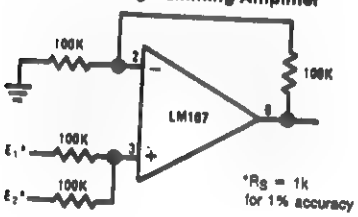
Difference Amplifier



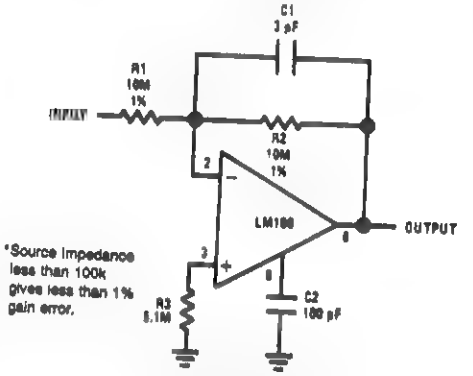
Inverting Summing Amplifier



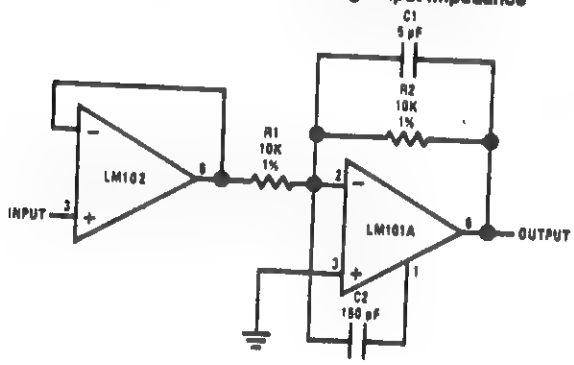
Non-Inverting Summing Amplifier



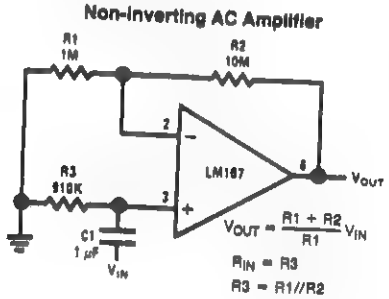
Inverting Amplifier with High Input Impedance



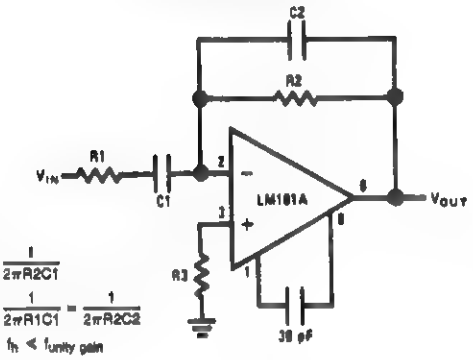
Fast Inverting Amplifier with High Input Impedance



Non-Inverting AC Amplifier



Practical Differentiator

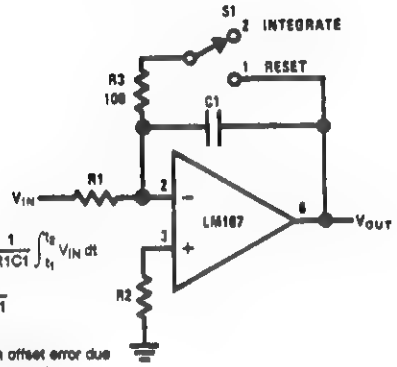


$f_o = \frac{1}{2\pi R2C1}$

$f_h = \frac{1}{2\pi R1C1} = \frac{1}{2\pi R2C2}$

$f_o < f_h < f_{unity\ gain}$

Integrator



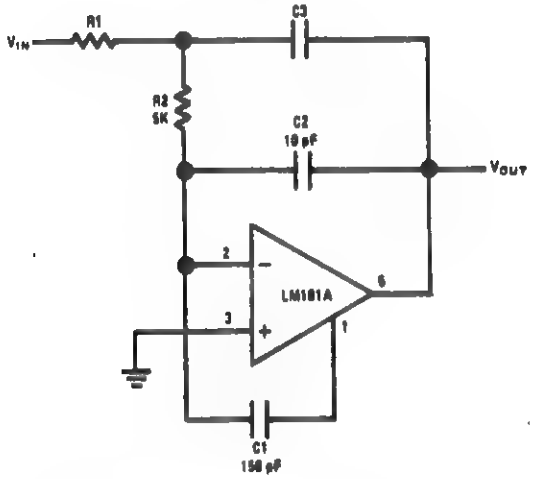
$V_{OUT} = -\frac{1}{R1C1}\int_{t_1}^{t_2} V_{IN}\,dt$

$f_o = \frac{1}{2\pi R1C1}$

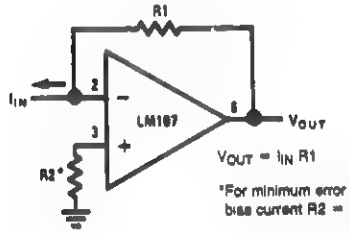
R1 = R2

For minimum offset error due to input bias current

Fast Integrator



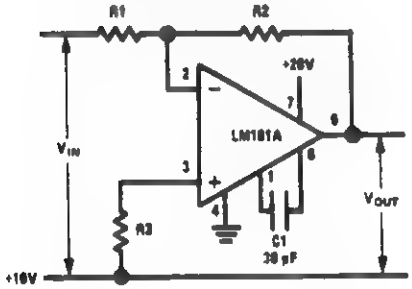
Current to Voltage Converter



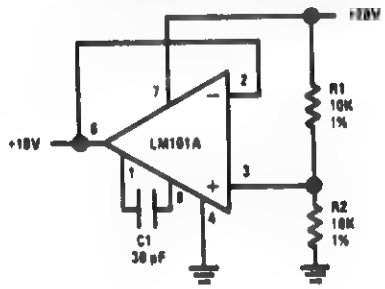
$V_{OUT} = I_{IN}\,R1$

*For minimum error due to bias current R2 = R1

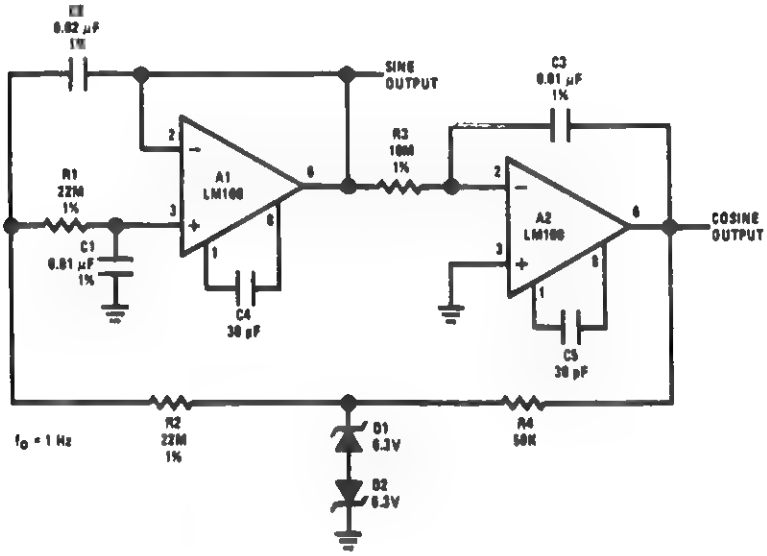
Circuit for Operating the LM101 without a Negative Supply



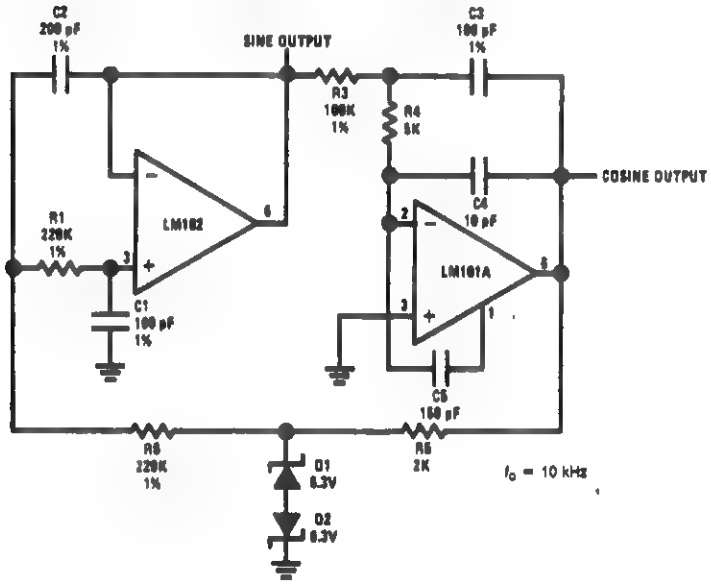
Circuit for Generating the Second Positive Voltage



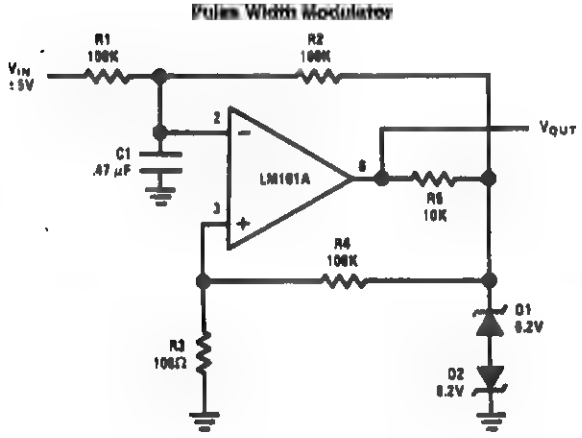
Low Frequency Sine Wave Generator with Quadrature Output



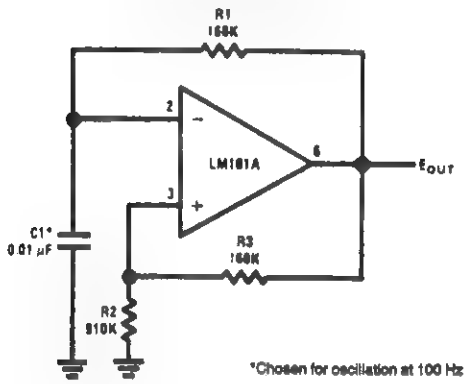
High Frequency Sine Wave Generator with Quadrature Output



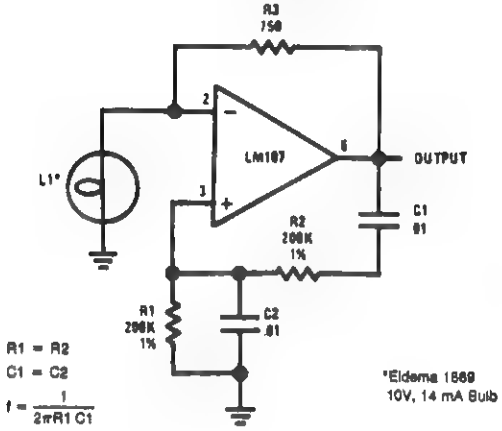
Pulse Width Modulator



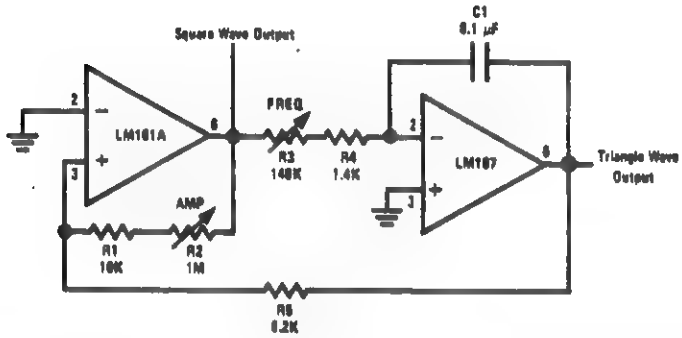
Free-Running Multivibrator



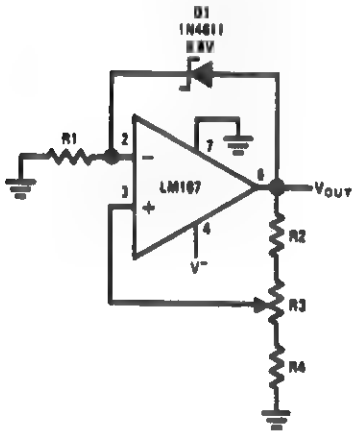
Wein Bridge Sine Wave Oscillator



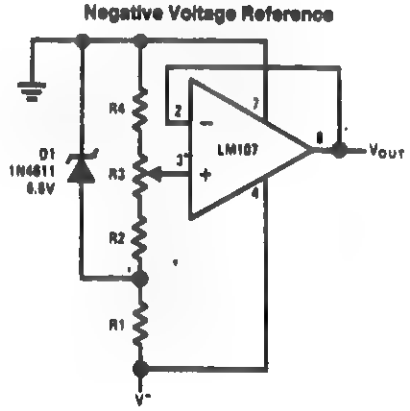
Function Generator



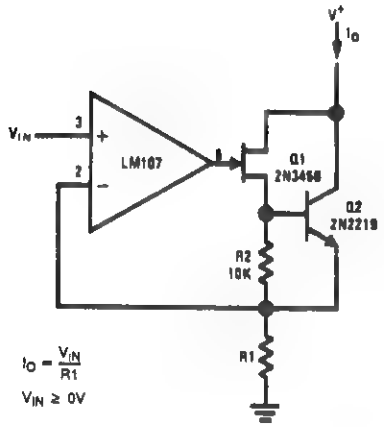
Negative Voltage Reference



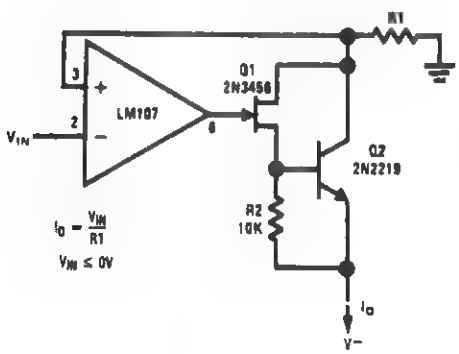
Negative Voltage Reference



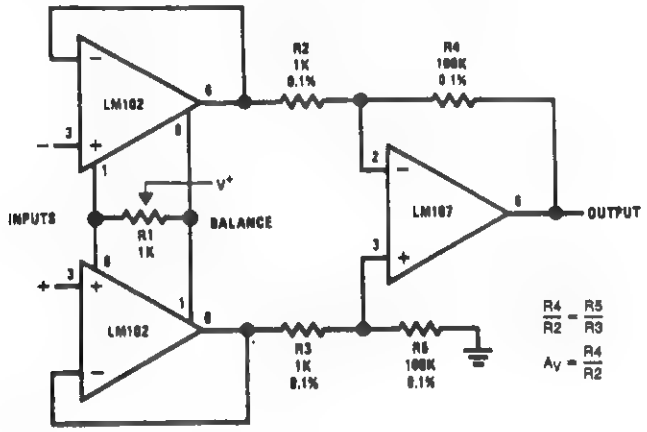
Precision Current Sink



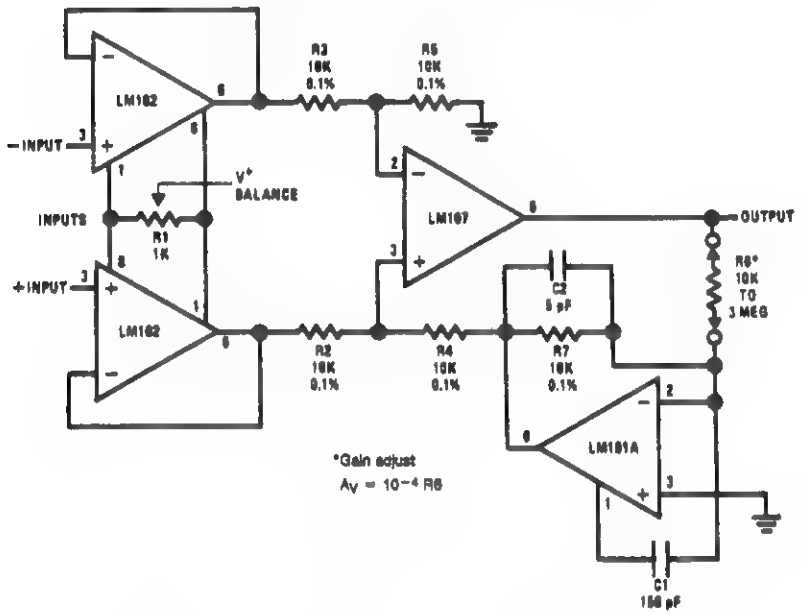
Precision Current Source



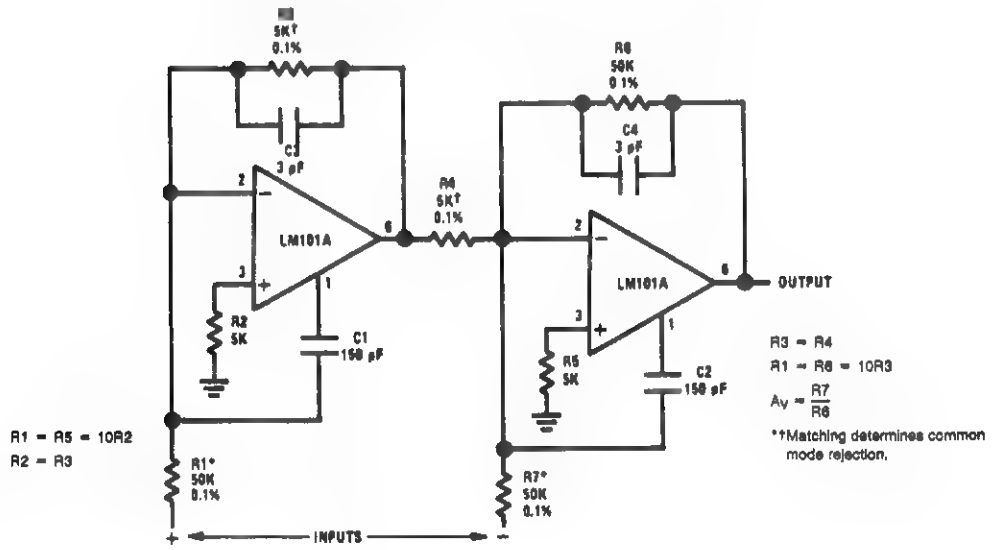
Differential-Input Instrumentation Amplifier



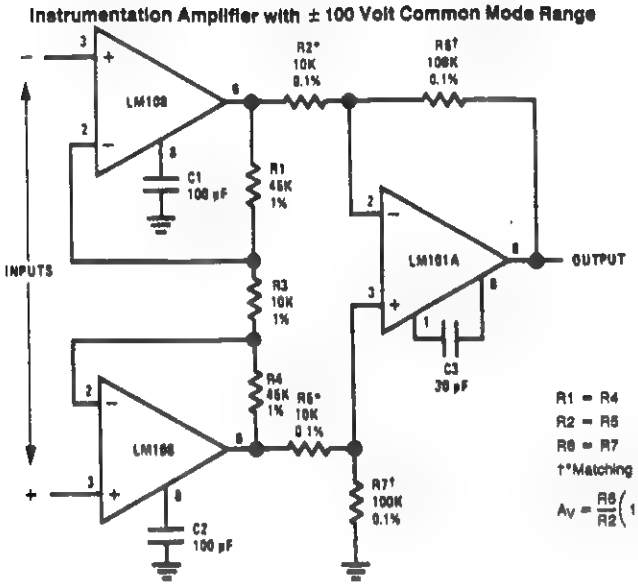
Variable Gain, Differential-Input Instrumentation Amplifier



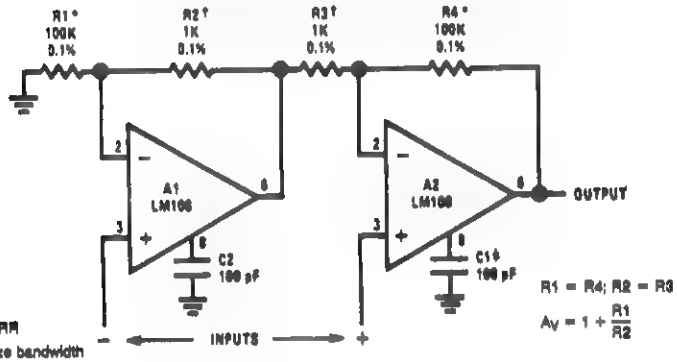
Instrumentation Amplifier with ±100 Volt Common Mode Range



Instrumentation Amplifier with ± 100 Volt Common Mode Range

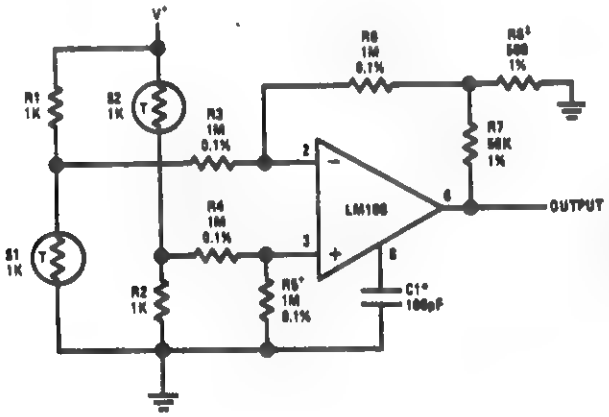


High Input Impedance Instrumentation Amplifier



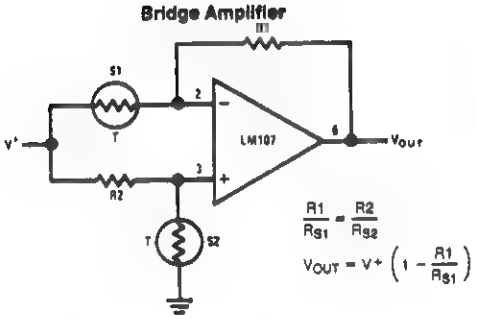
*†Matching determines CMRR
‡May be deleted to maximize bandwidth

Bridge Amplifier with Low Noise Compensation

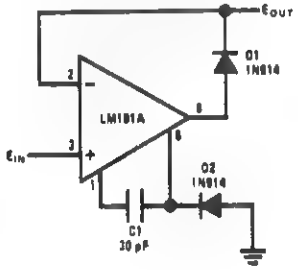


*Reduces feed through of power supply noise by 20 dB and makes supply bypassing unnecessary.
†Trim for best common mode rejection
‡Gain adjust

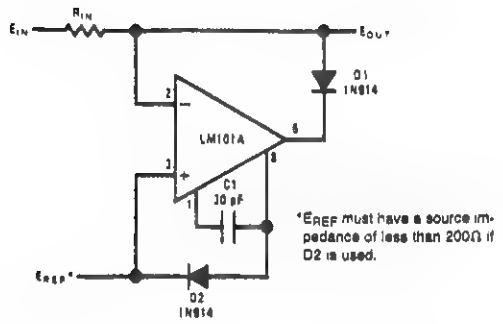
Bridge Amplifier



Precision Diode

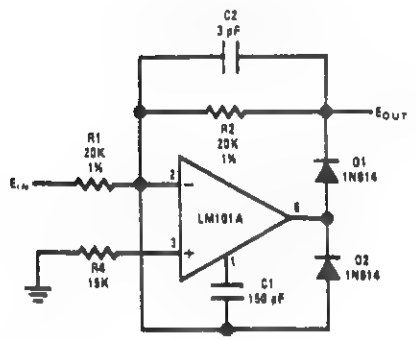


Precision Clamp

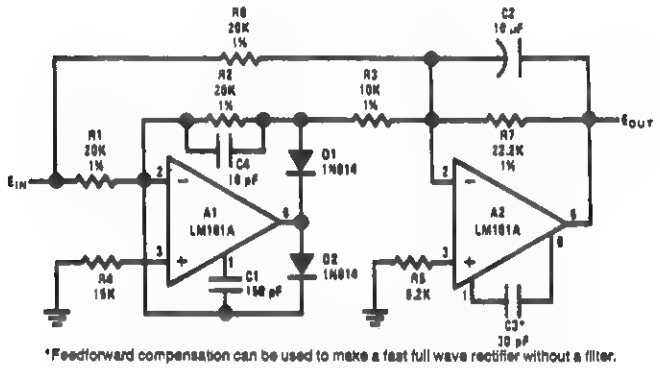


*E_{REF} must have a source impedance of less than 200Ω if D2 is used.

Fast Half Wave Rectifier

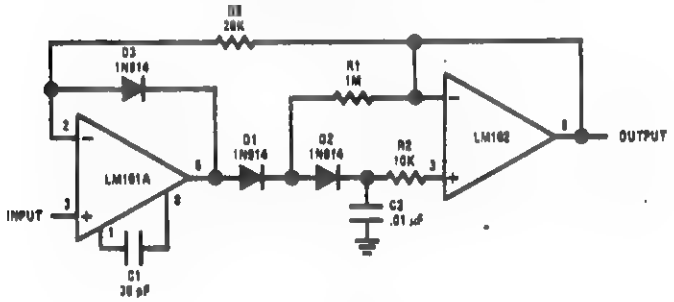


Precision AC to DC Converter

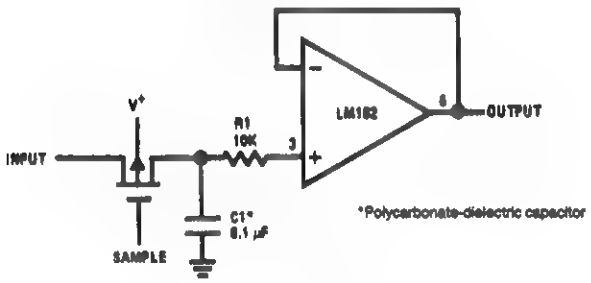


*Feedforward compensation can be used to make a fast full wave rectifier without a filter.

Low Drift Peak Detector

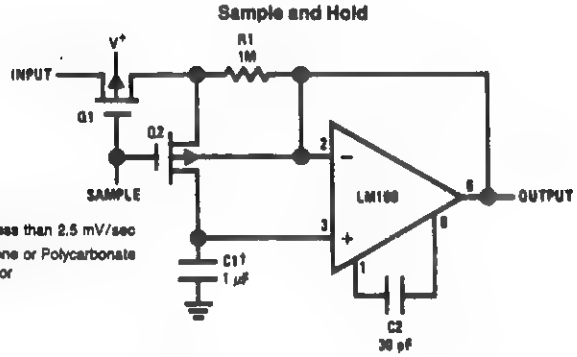


Sample and Hold



*Polycarbonate-dielectric capacitor

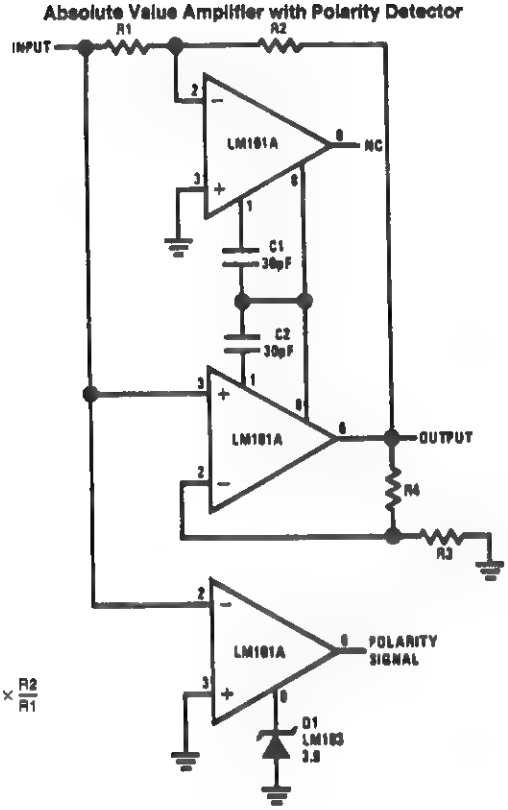
Sample and Hold



*Worst case drift less than 2.5 mV/sec

†Teflon, Polyethylene or Polycarbonate Dielectric Capacitor

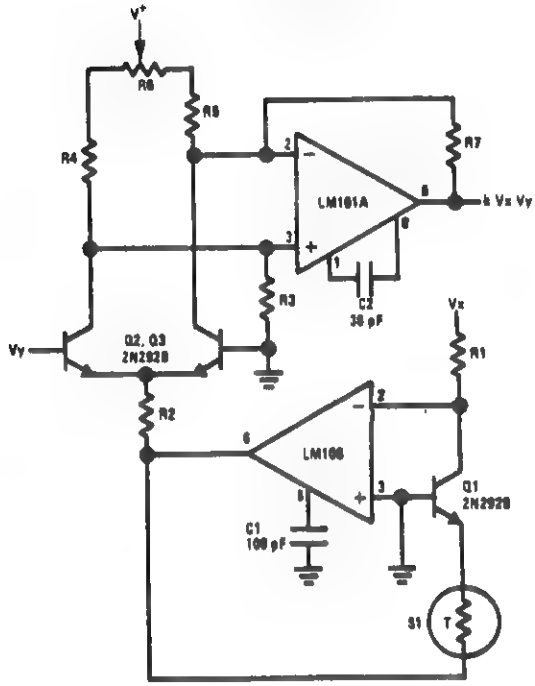
Absolute Value Amplifier with Polarity Detector



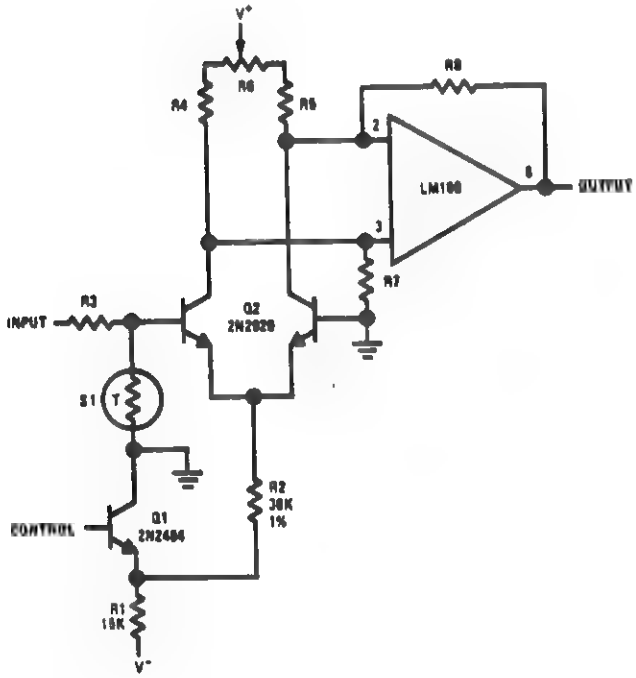
$$V_{OUT} = -|V_{IN}| \times \frac{R2}{R1}$$

$$\frac{R2}{R1} = \frac{R4 + R3}{R3}$$

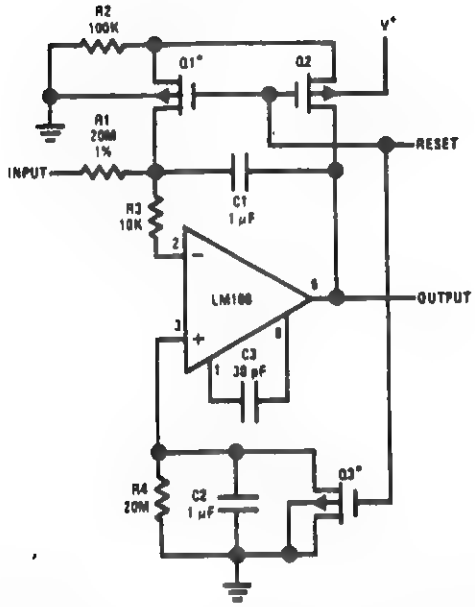
Two Quadrant Multiplier



Voltage Controlled Gain Circuit



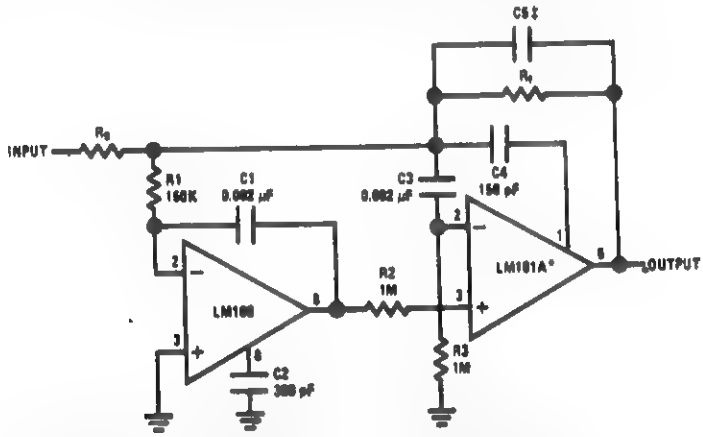
Low Drift Integrator



*Q1 and Q3 should not have internal gate-protection diodes.

Worst case drift less than 600 μV/sec over −55°C to +125°C.

Fast† Summing Amplifier with Low Input Current

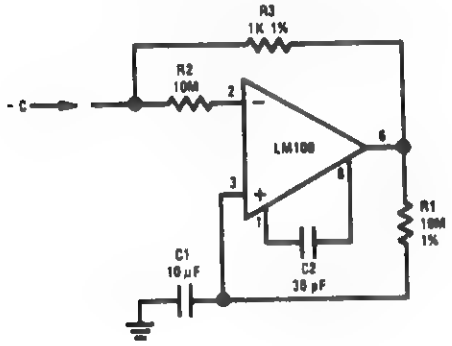


* In addition to increasing speed, the LM101A raises high and low frequency gain, increases output drive capability and eliminates thermal feedback.

† Power Bandwidth: 250 kHz
Small Signal Bandwidth: 3.5 MHz
Slew Rate: 10V/μs

‡ $C5 = \frac{6 \times 10^{-6}}{R_f}$

Negative Capacitance Multiplier

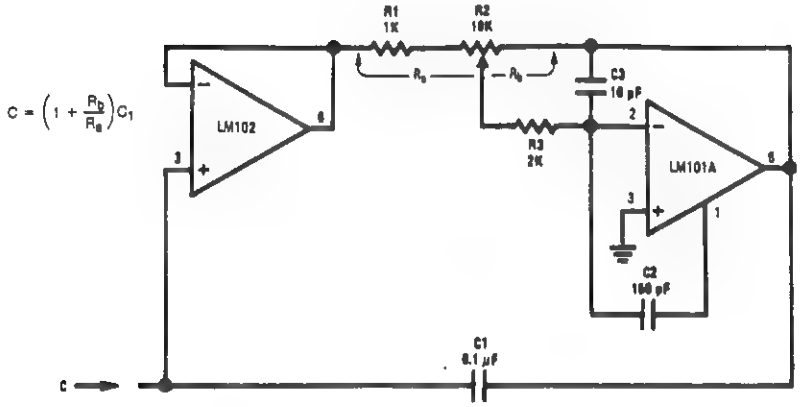


$$C = \frac{R2}{R3} C1$$

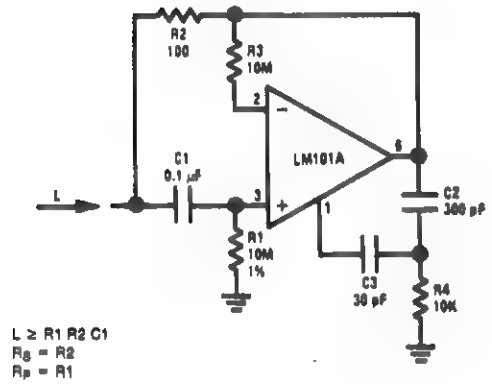
$$I_L = \frac{V_{OS} + R2\, I_{OS}}{R3}$$

$$R_S = \frac{R3(R1 + R_{IN})}{R_{IN} A_{VO}}$$

Variable Capacitance Multiplier

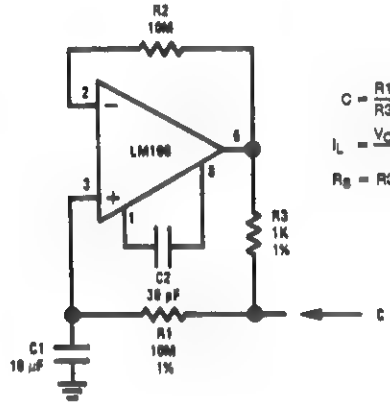


Simulated Inductor



$L \geq R1\ R2\ C1$
$R_S = R2$
$R_P = R1$

Capacitance Multiplier

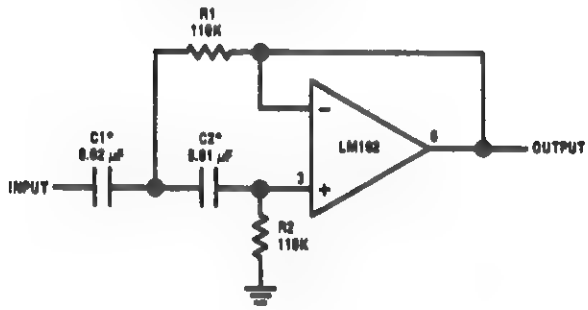


$C = \frac{R1}{R3}C1$

$I_L = \frac{V_{OS} + I_{OS}R1}{R3}$

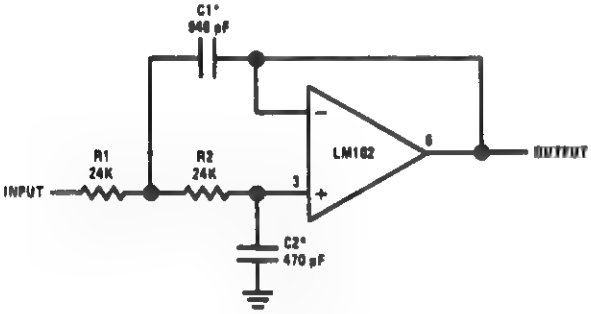
$R_S = R3$

High Pass Active Filter



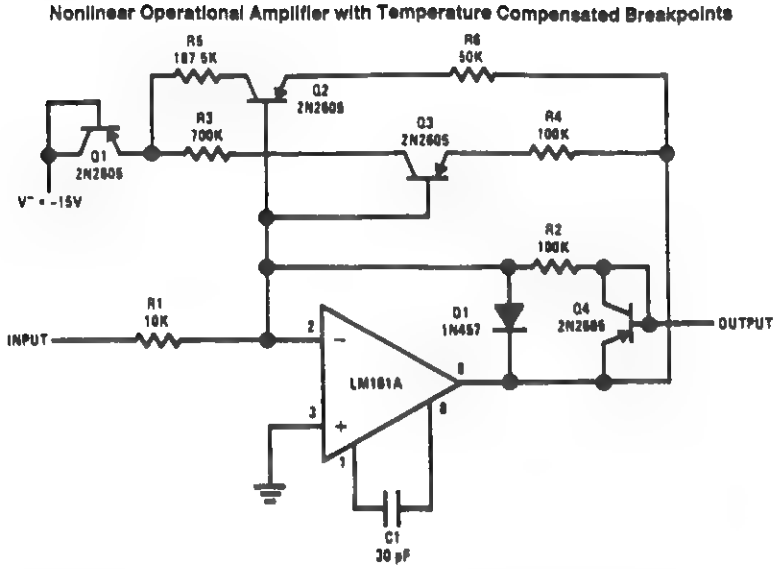
*Values are for 100 Hz cutoff. Use metalized polycarbonate capacitors for good temperature stability.

Low Pass Active Filter

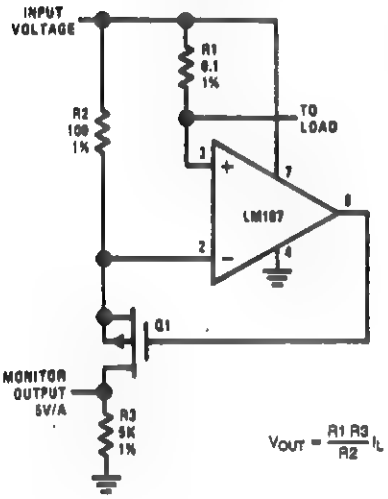


*Values are for 10 kHz cutoff. Use silvered mica capacitors for good temperature stability.

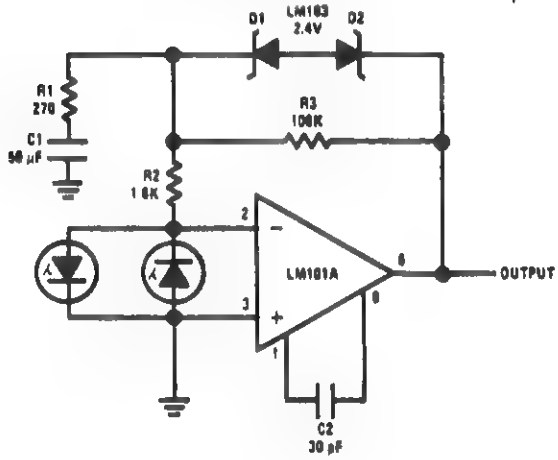
Nonlinear Operational Amplifier with Temperature Compensated Breakpoints



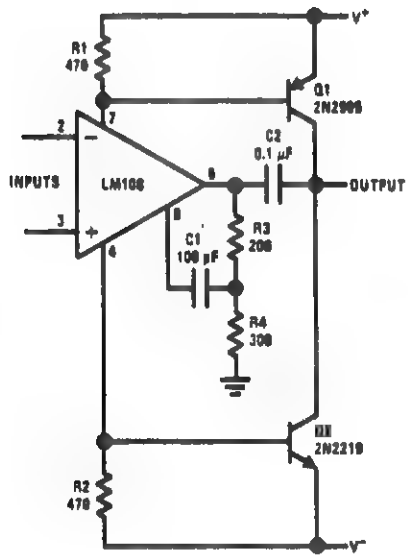
Current Monitor



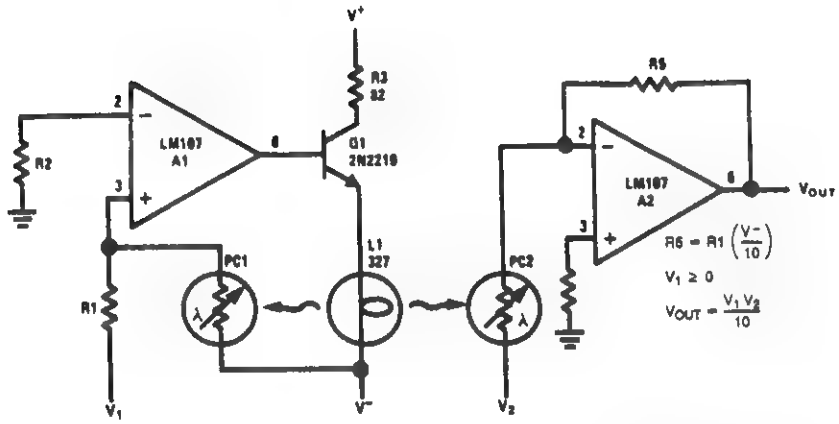
Saturating Servo Preamplifier with Rate Feedback



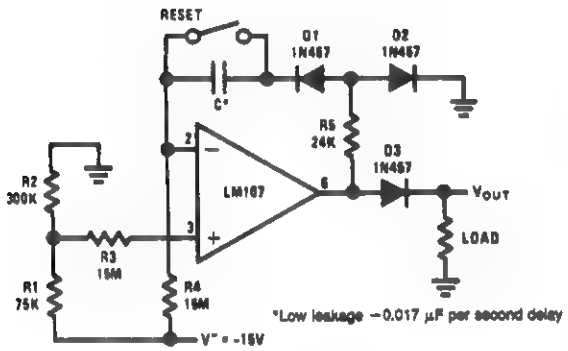
Power Booster



Analog Multiplier

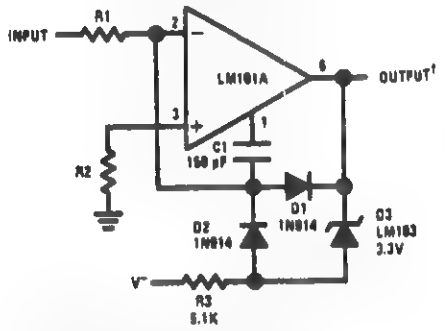


Long Interval Timer



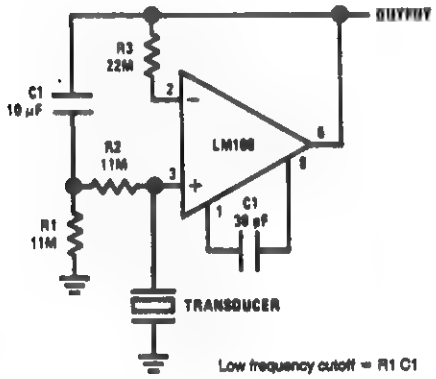
*Low leakage −0.017 μF per second delay

Fast Zero Crossing Detector



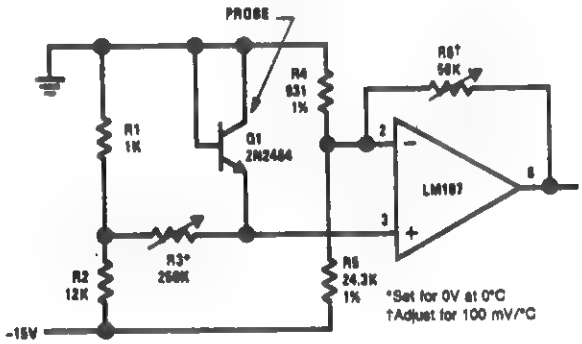
Propagation delay approximately 200 ns
†DTL or TTL fanout of three.
Minimize stray capacitance
Pin 8

Amplifier for Piezoelectric Transducer



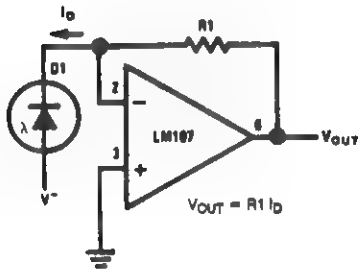
Low frequency cutoff = R1 C1

Temperature Probe

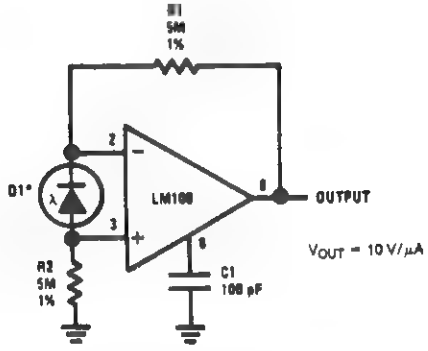


*Set for 0V at 0°C
†Adjust for 100 mV/°C

Photodiode Amplifier

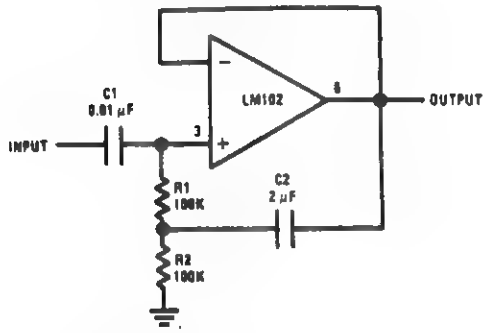


Photodiode Amplifier

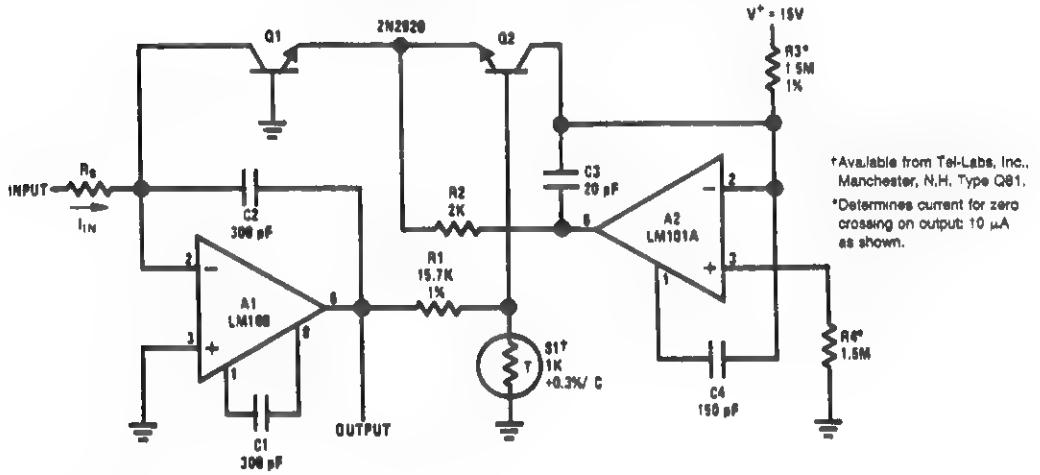


*Operating photodiode with less than 3 mV across it eliminates leakage currents.

High Input Impedance AC Follower



Temperature Compensated Logarithmic Converter

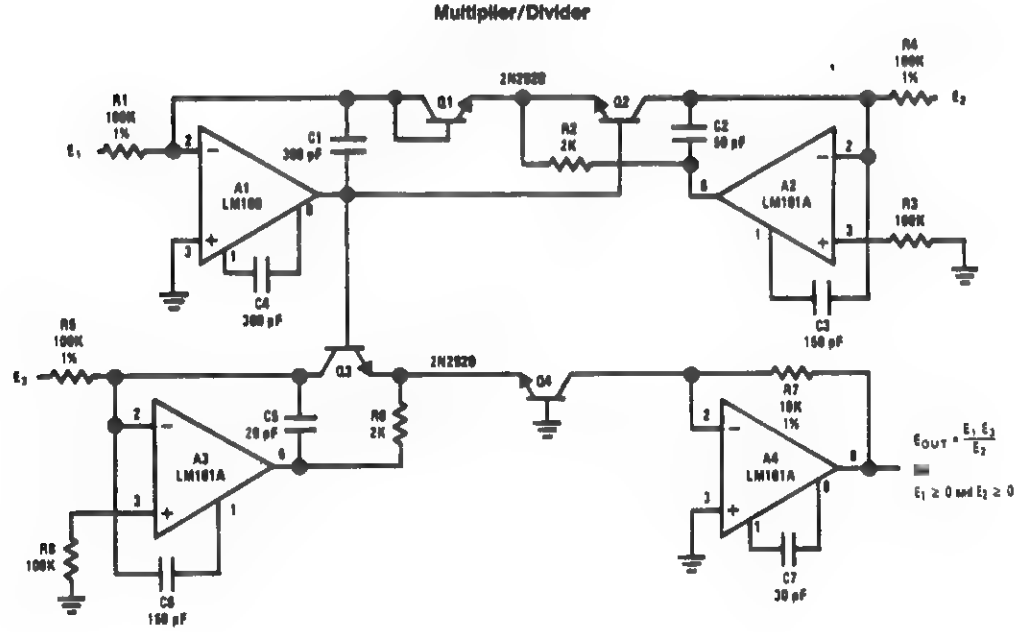


†Available from Tel-Labs, Inc., Manchester, N.H. Type Q81.

*Determines current for zero crossing on output: 10 µA as shown.

10 nA < I_{IN} < 1 mA
Sensitivity is 1V per decade

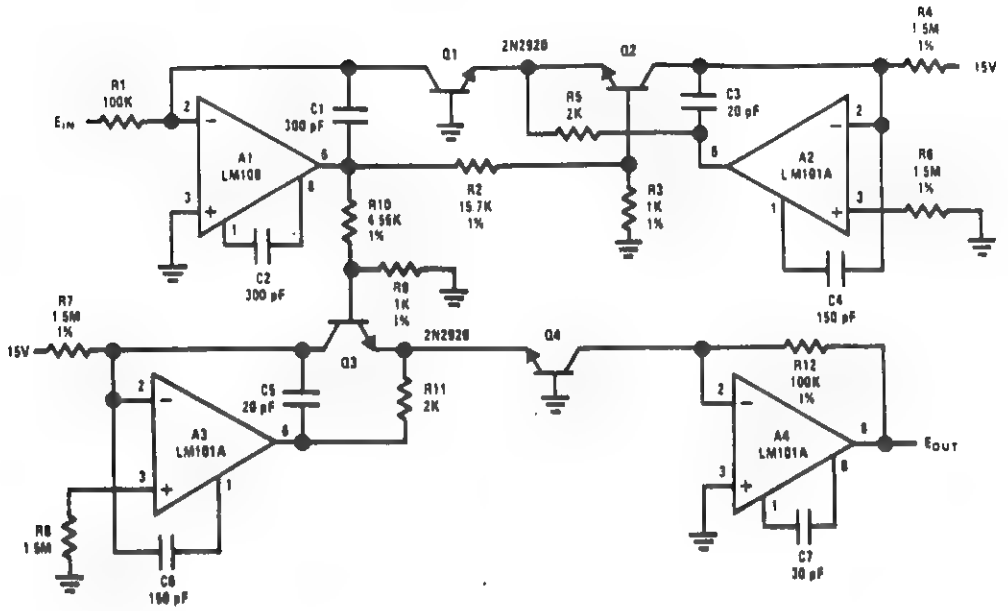
Multiplier/Divider



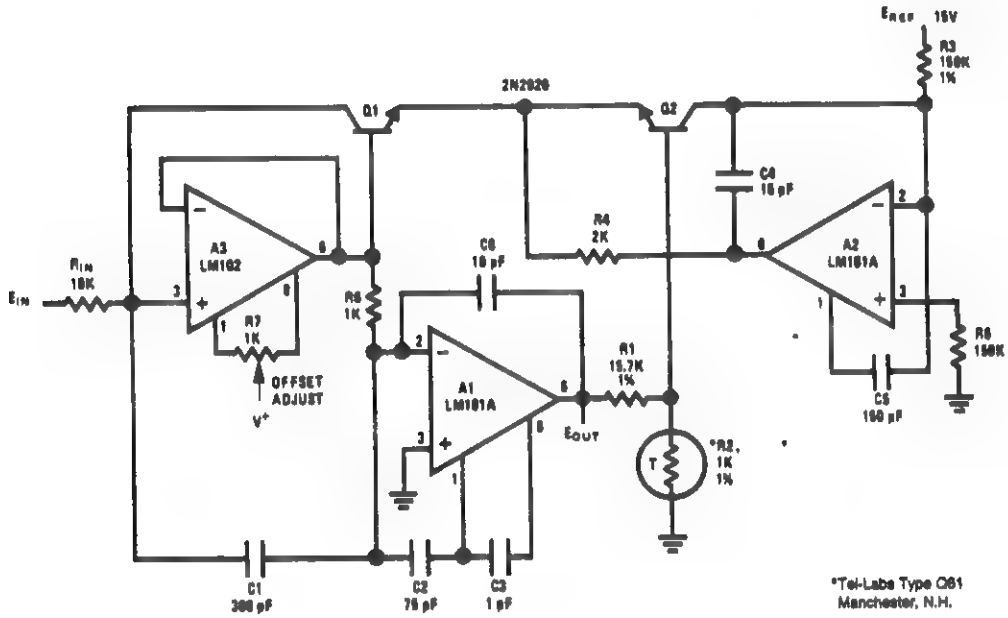
$E_{OUT} = \frac{E_1 E_2}{E_3}$

$E_1 \geq 0$ and $E_2 \geq 0$

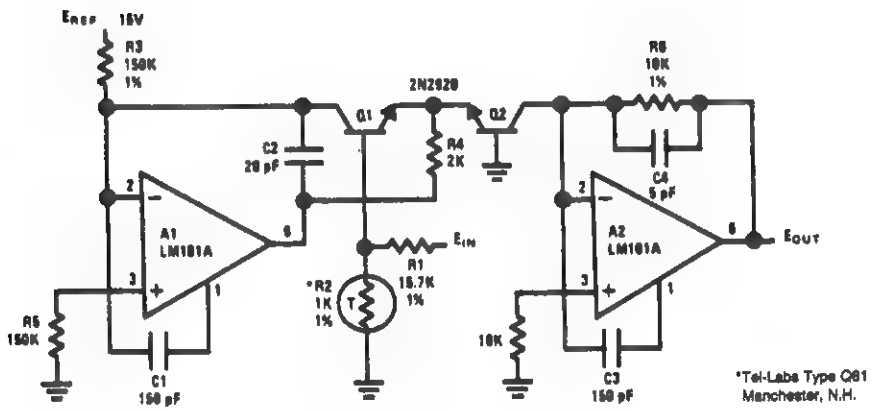
Cube Generator

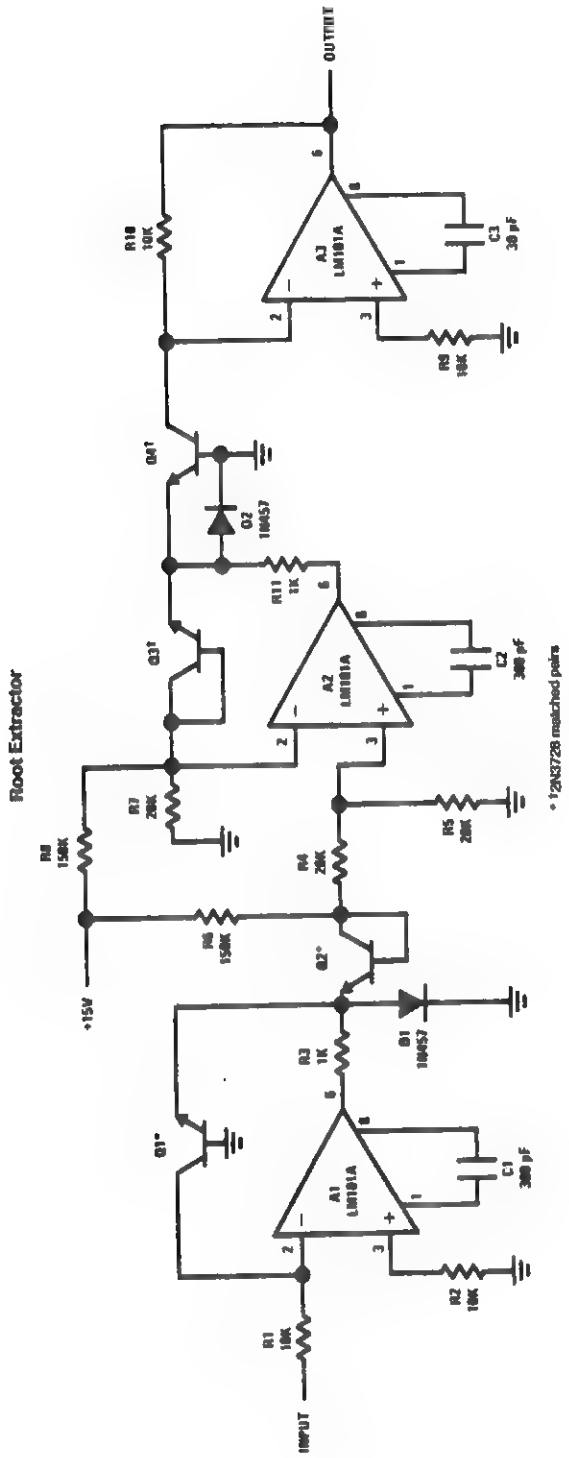


Fast Log Generator



Anti-Log Generator




Root Extractor
OUTPUT
INPUT
+15V
A1 LM101A
A2 LM101A
A3 LM101A
C1 300 pF
C2 300 pF
C3 30 pF
R1 10K
R2 10K
R3 1K
R4 20K
R5 20K
R6 150K
R7 20K
R8 150K
R9 10K
R11 1K
D1 1N457
D2 1N457
Q1* matched pairs
Q2* matched pairs
Q3* matched pairs
Q4* matched pairs
* 1/2N3728 matched pairs

Annex 5
INTERFÍCIE DIGITAL D'ENTRADES/SORTIDES

1.- INTRODUCCIÓ.

La tarja 8255 I/O és una interfície d'entrades/sortides programable per a P.C. Aquesta tarja conté 48 línies d'entrada/sortida i tres comptadors independents, de 16 bits cada un, amb una velocitat màxima de 2 MHz. Totes les modalitats d'operació són programables per software.

Per obtenir els 48 bits d'E/S utilitza dos xips Intel 8255, que proporcionen 3 ports de 8 bits d'E/E cada xip, i un xip Intel 8253, que proporciona els tres comptadors. Els ports de cada xip es poden programar com a ports d'entrada/sortida, ports d'entrada/sortida amb protocol o ports bidireccionals.

2.- ADRECES DELS PORTS.

Les adreces dels ports es poden configurar amb uns microinterruptors situats en la mateixa placa.

SW4 = OFF , SW5 = ON → Selecciona els ports $1B0 - $1BF
SW4 = ON , SW5 = OFF → Selecciona els ports $1F0 - $1FF

Per defecte, l'adreça dels ports és: $1B0 - $1BF

$1B0 → Port 1A, buffer de lectura/escriptura
$1B1 → Port 1B, buffer de lectura/escriptura
$1B2 → Port 1C, buffer de lectura/escriptura
$1B3 → Port 1 , registre de control (1r. 8255)
$1B4 → Port 2A, buffer de lectura/escriptura
$1B5 → Port 2B, buffer de lectura/escriptura
$1B6 → Port 2C, buffer de lectura/escriptura
$1B7 → Port 2 , registre de control (2n. 8255)
$1B8 → Comptador 0, buffer de lectura/escriptura
$1B9 → Comptador 1, buffer de lectura/escriptura
$1BA → Comptador 2, buffer de lectura/escriptura
$1BB → Registre de control del xip 8253

3.- "PIN OUT" DELS CONNECTORS.

El "pin out" dels dos connectors situats en la mateixa placa 8255 I/O està indicat en la figura 1. El primer connector "CN 1 (PORT 1)" conté els 3 ports d'E/S del 1r xip 8255, i els tres comptadors del xip 8253, així com les tensions de +5V, -5V, +12V, -12V i GND extretes de l'ordinador. El segon connector "CN 2 (PORT 2)" conté els tres ports d'E/S del segon xip 8255.

A la placa 8255 I/O se li ha connectat una tarja exterior que suporta totes les connexions del primer xip 8255 i del comptador 8253. El pin-out d'aquesta tarja externa és el mateix que del connector intern "CN 1 (PORT 1)" de la tarja 8255 I/O.

CN 1 (PORT 1)

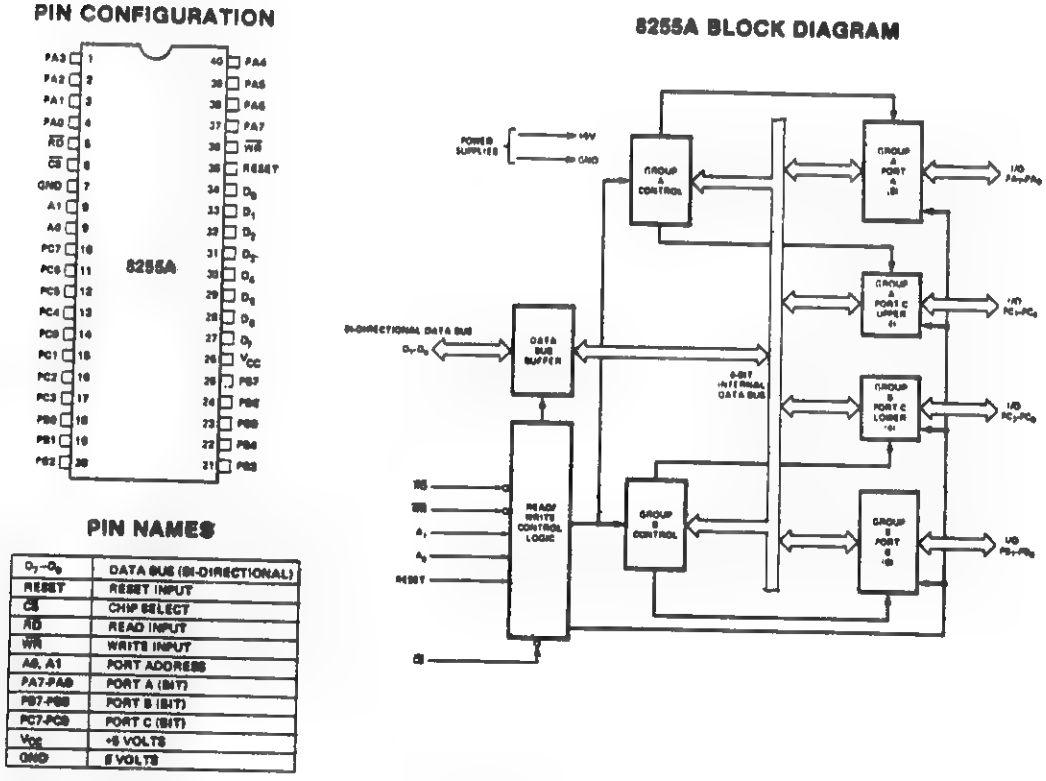
PIN			PIN	
	1	GND	21	PC6
	2	GND	22	PC7
	3	PA3	23	PC5
	4	NC	24	PC4
	5	PA2	25	PC0
	6	PA1	26	PC1
	7	PA0	27	PC2
	8	CLKO	28	PB7
	9	OUTO	29	PC3
	10	GATEO	30	PB6
	11	CLK2	31	PB0
	12	OUT2	32	PB5
	13	GATE2	33	PB1
	14	CLK1	34	PB4
	15	GATE1	35	PB2
	16	OUT 1	36	PB3
	17	PA4	37	−5V
	18	PA5	38	+5V
	19	PA6	39	−12V
	20	PA7	40	+12V

CN 2 (PORT 2)

PIN			PIN	
	1	GND	21	PC6
	2	GND	22	PC7
	3	NC	23	PC4
	4	NC	24	PC5
	5	NC	25	PC1
	6	NC	26	PC0
	7	NC	27	PB 7
	8	NC	28	PC2
	9	NC	29	PB6
	10	NC	30	PC3
	11	NC	31	PB5
	12	NC	32	PB0
	13	PA1	33	PB4
	14	PA0	34	PB1
	15	PA3	35	PB3
	16	PA2	36	PB2
	17	PA5	37	−5V
	18	PA4	38	+5V
	19	PA7	39	−12V
	20	PA6	40	+12V

Figura 1. Llistat dels pins de sortida de la placa 8255 I/O.

4.- DIAGRAMA DE BLOCS DEL XIP INTEL 8255.



D_7–D_0	DATA BUS (BI-DIRECTIONAL)
RESET	RESET INPUT
$\overline{CS}$	CHIP SELECT
$\overline{RD}$	READ INPUT
$\overline{WR}$	WRITE INPUT
A0, A1	PORT ADDRESS
PA7-PA0	PORT A (BIT)
PB7-PB0	PORT B (BIT)
PC7-PC0	PORT C (BIT)
V_{CC}	+5 VOLTS
GND	0 VOLTS

Figura 2. Diagrama de blocs del xip 8255.

5.- FUNCIONAMENT DEL XIP INTEL 8255.

El xip 8255 té tres modes bàsics de funcionament, que es poden seleccionar per software:

Mode 0 → Modalitat bàsica d'Entrades/Sortides
Mode 1 → Modalitat d'Entrades/Sortides amb protocol
Mode 2 → Modalitat Bus bidireccional

La modalitat que ens interessa per treballar com a simples ports digitals d'entrada/sortida és el mode 0.

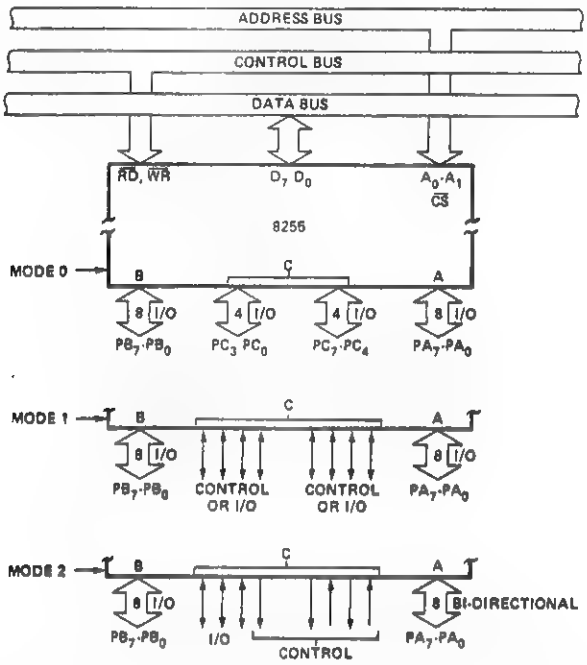


Figura 3. Definició dels modes de treball i connexionat amb el bus.

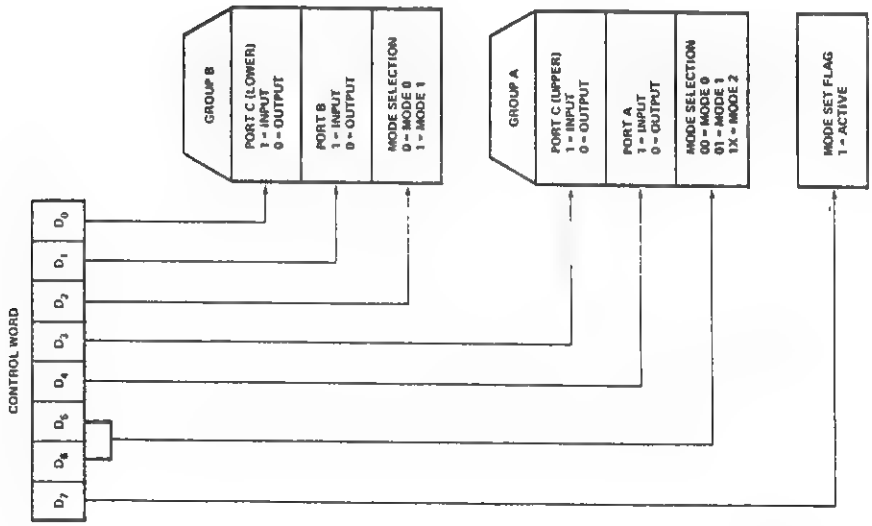
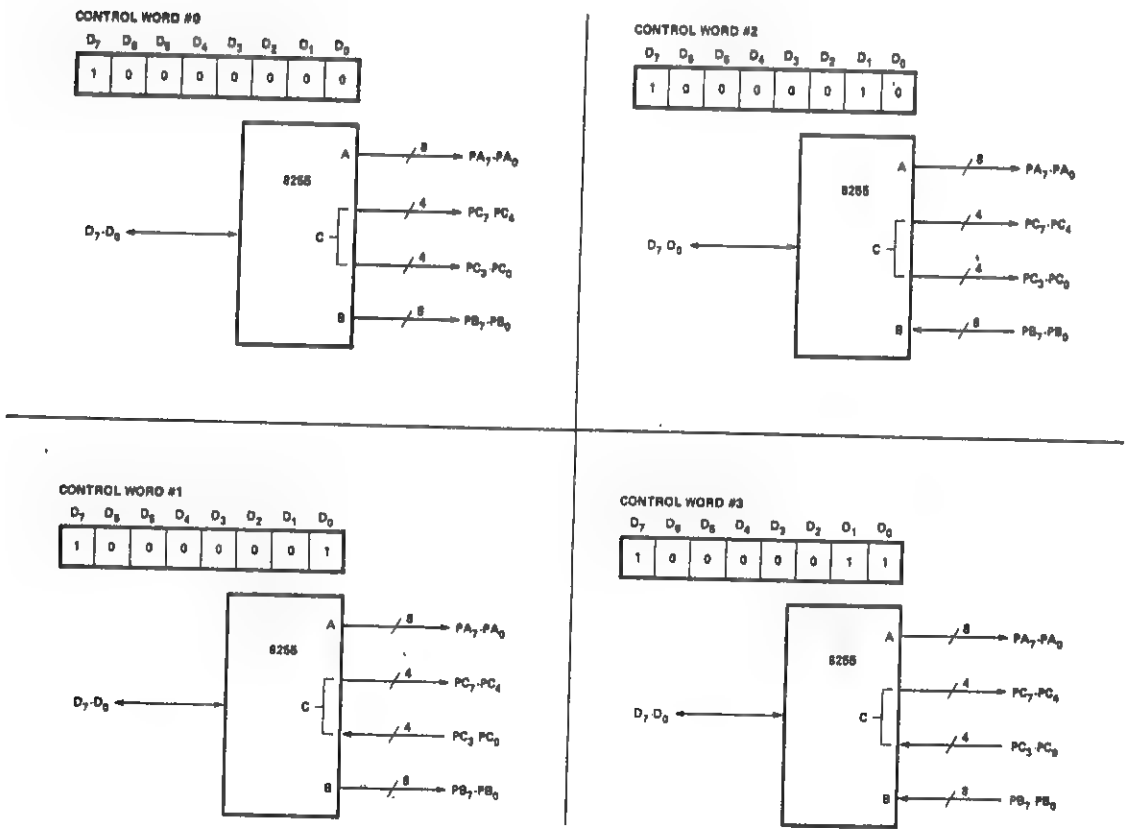


Figura 4. Format del registre que defineix el mode de funcionament.

A		B		GROUP A			GROUP B	
D_4	D_3	D_1	D_0	PORT A	PORT C (UPPER)	#	PORT B	PORT C (LOWER)
0	0	0	0	OUTPUT	OUTPUT	0	OUTPUT	OUTPUT
0	0	0	1	OUTPUT	OUTPUT	1	OUTPUT	INPUT
0	0	1	0	OUTPUT	OUTPUT	2	INPUT	OUTPUT
0	0	1	1	OUTPUT	OUTPUT	3	INPUT	INPUT
0	1	0	0	OUTPUT	INPUT	4	OUTPUT	OUTPUT
0	1	0	1	OUTPUT	INPUT	5	OUTPUT	INPUT
0	1	1	0	OUTPUT	INPUT	6	INPUT	OUTPUT
0	1	1	1	OUTPUT	INPUT	7	INPUT	INPUT
1	0	0	0	INPUT	OUTPUT	8	OUTPUT	OUTPUT
1	0	0	1	INPUT	OUTPUT	9	OUTPUT	INPUT
1	0	1	0	INPUT	OUTPUT	10	INPUT	OUTPUT
1	0	1	1	INPUT	OUTPUT	11	INPUT	INPUT
1	1	0	0	INPUT	INPUT	12	OUTPUT	OUTPUT
1	1	0	1	INPUT	INPUT	13	OUTPUT	INPUT
1	1	1	0	INPUT	INPUT	14	INPUT	OUTPUT
1	1	1	1	INPUT	INPUT	15	INPUT	INPUT

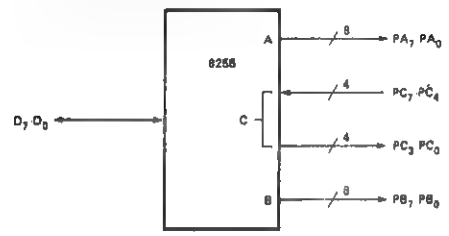
Figura 5. Definició dels ports en el Mode 0.

Figura 6. Configuracions del Mode 0.



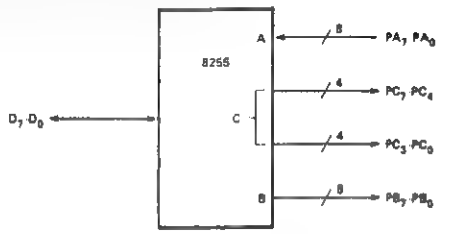
CONTROL WORD #4

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
1	0	0	0	1	0	0	0



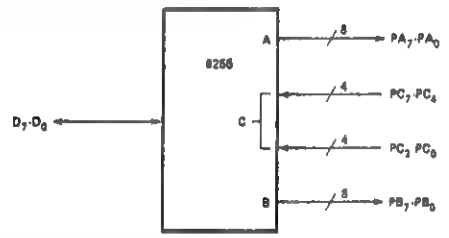
CONTROL WORD #8

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
1	0	0	1	0	0	0	0



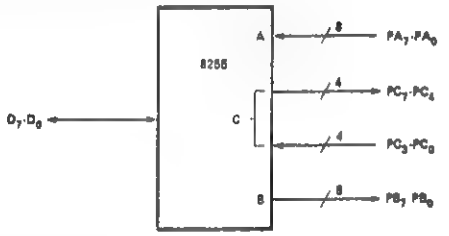
CONTROL WORD #5

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
1	0	0	0	1	0	0	1



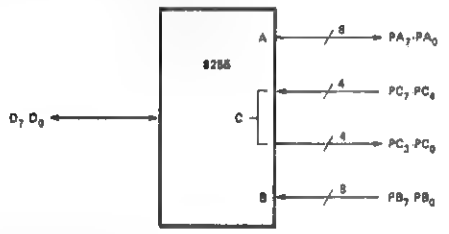
CONTROL WORD #9

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
1	0	0	1	0	0	0	1



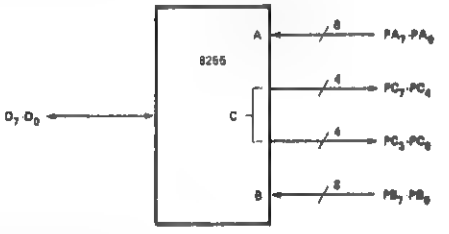
CONTROL WORD #6

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
1	0	0	0	1	0	1	0



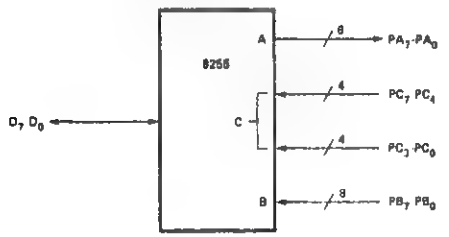
CONTROL WORD #10

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
1	0	0	1	0	0	1	0



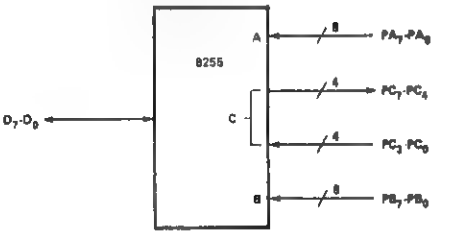
CONTROL WORD #7

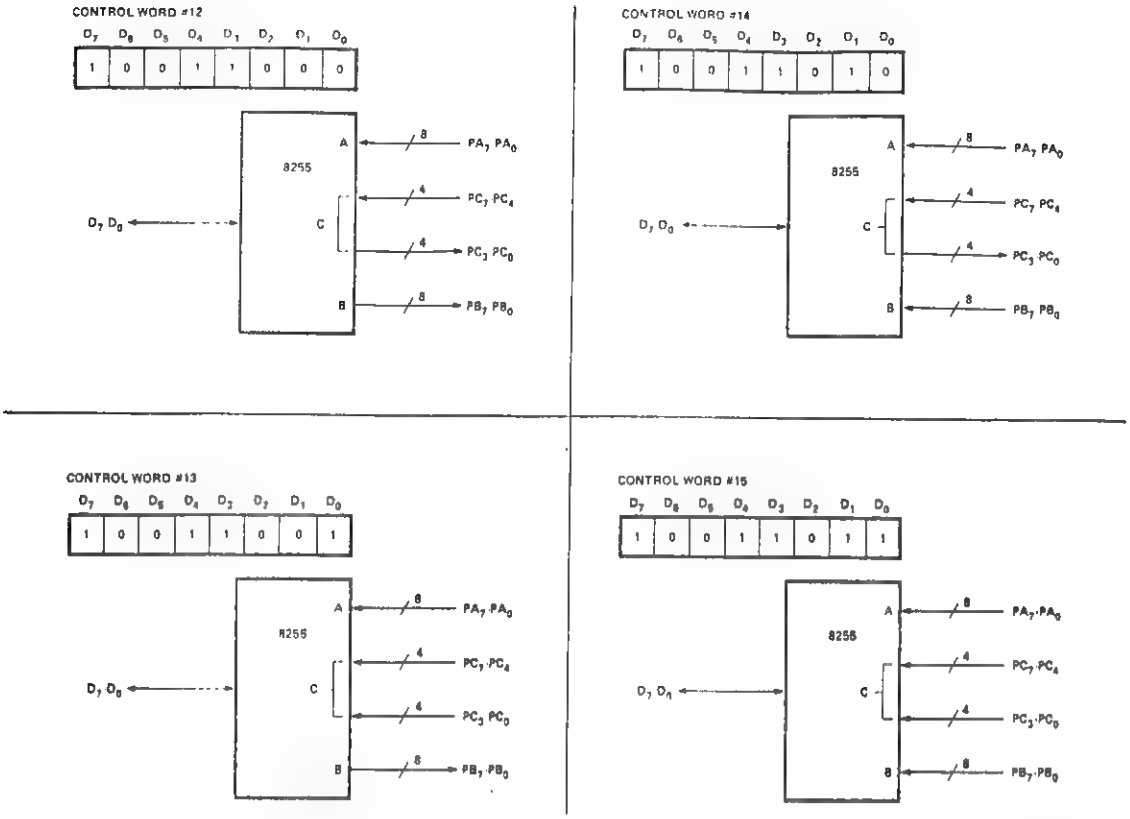
D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
1	0	0	0	1	0	1	1



CONTROL WORD #11

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
1	0	0	1	0	0	1	1





ABSOLUTE MAXIMUM RATINGS*

Ambient Temperature Under Bias. . . . . 0°C to 70°C
Storage Temperature . . . . . –65°C to +150°C
Voltage on Any Pin
With Respect to Ground . . . . . –0.5V to +7V
Power Dissipation . . . . . 1 Watt

**COMMENT: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.*

D.C. CHARACTERISTICS T_A = 0°C to 70°C, V_{CC} = +5V ±5%; GND = 0V

SYMBOL	PARAMETER	MIN.	MAX.	UNIT	TEST CONDITIONS
V_{IL}	Input Low Voltage	-0.5	0.8	V	
V_{IH}	Input High Voltage	2.0	V_{CC}	V	
V_{OL} (DB)	Output Low Voltage (Data Bus)		0.45	V	I_{OL} = 2.5mA
V_{OL}(PER)	Output Low Voltage (Peripheral Port)		0.45	V	I_{OL} = 1.7mA
V_{OH}(DB)	Output High Voltage (Data Bus)	2.4		V	I_{OH} = -400μA
V_{OH}(PER)	Output High Voltage (Peripheral Port)	2.4		V	I_{OH} = -200μA
I_{DAR}(1)	Darlington Drive Current	-1.0	-4.0	mA	R_{EXT} = 750Ω; V_{EXT} = 1.5V
I_{CC}	Power Supply Current		120	mA	
I_{IL}	Input Load Current		±10	μA	V_{IN} = V_{CC} to 0V
I_{OFL}	Output Float Leakage		±10	μA	V_{OUT} = V_{CC} to 0V

Note 1: Available on any 8 pins from Port B and C.

CAPACITANCE T_A = 25°C; V_{CC} = GND = 0V

SYMBOL	PARAMETER	MIN.	TYP.	MAX.	UNIT	TEST CONDITIONS
C_{IN}	Input Capacitance			10	pF	fc = 1MHz
$C_{I/O}$	I/O Capacitance			20	pF	Unmeasured pins returned to GN

Figura 7. Característiques en C.C.

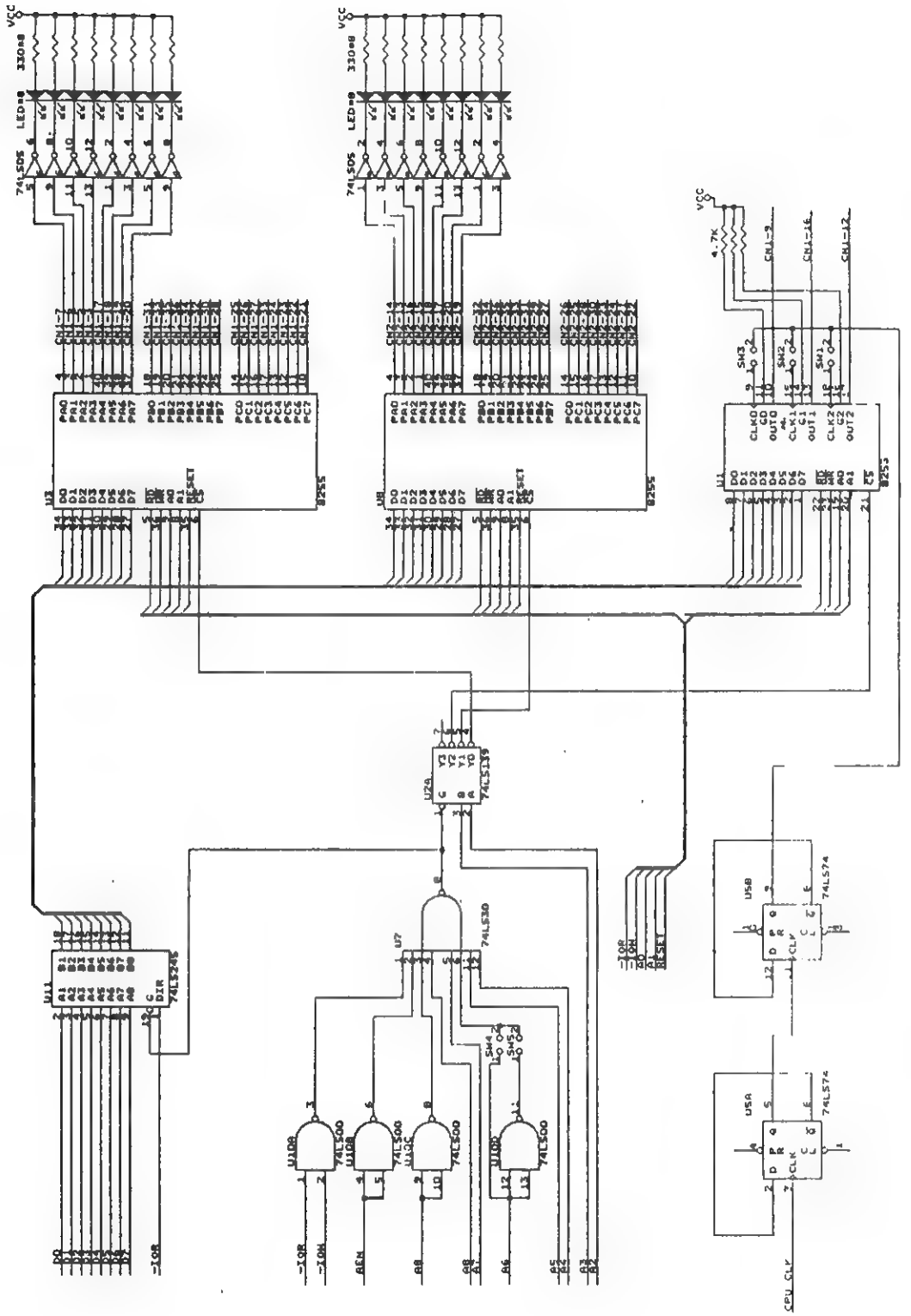


Figura 8. Circuit de la tarja 8255 I/O.

Annex 6

MANUAL DE LA TARJA D'ADQUISICIÓ DE DADES PCL-711B

Model PCL-711B

PC-MultiLab Card

TABLE OF CONTENTS

CHAPTER 1. INTRODUCTION

The PCL-711B PC-MultiLab Card is an easy-to-use and cost/effective IBM PC/XT/AT compatible multifunction data acquisition card. This card's specifications and user-friendly software driver make it a popular solution for a wide range of industrial and laboratory applications. Such applications include: data acquisition, process control, automatic testing, and factory automation.

1.1. Features

- 12-bit resolution A/D conversion
- Accepts 8 single-ended analog inputs
- Programmable analog input ranges: ±5V, ±2.5V, ±1.25, ±0.625V, ±0.3125V
- Support software trigger, programmable pacer trigger, and external trigger
- Programmable IRQ level for A/D data transfer
- One 12-bit multiplying D/A output channel, with the output range of 0 to +5V or 0 to +10V
- On-board 16-bit digital input and digital output
- Versatile language drivers including BASIC, PASCAL, C and C++

1.2. Specifications

1.2.1. Analog Input (A/D Converter)

Channels: 8 single-ended inputs.

Resolution: 12 bits, successive approximation.

Input range: ±5V, ±2.5V, ±1.25, ±0.625V, and ±0.3125V, software programmable.

Converter:	AD574 or equivalent
Conversion time:	25μs max.
Accuracy:	0.015% of reading ±1 LSB
Nonlinearity:	±1 bit
Amplification gains:	x1, x2, x4, x8, and x16, software programmable
Trigger mode:	By software, pacer and external trigger.
Data transfer:	By software or interrupt.
Overvoltage:	Continuous ±30V max.
IRQ level:	IRQ2 to IRQ7

1.2.2. Analog Output (D/A Converter)

Channels:	One channel.
Resolution:	12 bits
Output range:	0 to +5V or 0 to +10V.
Settling time:	30μs.
Reference voltage:	Internal -5V and -10V (±0.05V).
Converter:	PM7548GP or equivalent.
Nonlinearity:	±½ LSB.
Output capacity:	±5mA max.

1.2.3. Digital Input

Channels:	16 bits, TTL compatible
Input voltage:	Low - 0.8V max High - 2.0V min.
Input load:	Low - 0.4mA max @0.5V High - 0.05mA max @2.7

1.2.4. Digital Output

Channel:	16 bits, TTL compatible
Output voltage:	Low (sink): 8mA @0.5V max High (source): 0.4mA @2.4V min.

1.2.5. General Specifications

Power consumption:
+5V: 100mA, typical; 500mA max.
+12V: 40mA, typical; 100mA max.
-12V: 20mA, typical; 50mA max.

I/O connector:
- One 20-pin connector for A/D and D/A
- One 20-pin connector for digital input
- One 20-pin connector for digital output

I/O ports: requires 16 consecutive I/O ports per card

Operating temperature: 0 to 50°C (32 to 122°F).

Storage temperature: -20 to 65°C (-4 to 149°F).

Weight: 127 gm (4.49 oz.).

CHAPTER 2. INSTALLATION

2.1. Initial Inspection

The PCL-711B was thoroughly inspected before being shipped to you. Before installing the card into your PC, make sure that everything has been included with the package. You should also inspect the card for any defects or damages that may have occurred during shipment. If you find anything missing, defective or damaged, contact your PC-LabCard dealer immediately.

Here is a list of the materials included with your PCL-711B package:

- One PCL-711B PC-MultiLab Card
- One User's Manual
- One utility diskette which includes the card's software driver

In the PCL-711S package, two additional accessories are also included:

- One PCLD-7115 Wiring Terminal Board
- One 1 meter cable

2.2. Switch and Jumper Settings

The PCL-711B has been designed with ease-of-use in mind. On board the card you will notice that there is only one DIP switch (SW1), and only one set of jumper pins (JP1). These are used to set the PCL-711B's base address, and to select its D/A output voltage range. The following sub-sections go into this in more detail.

2.2.1. I/O Address Selection

Most peripheral devices and interface cards are controlled via your PC's I/O ports. These devices and cards should be placed in an appropriate I/O space so that there will be no conflicts between them and the PCL-711B. Keep in mind that the PCL-711B uses 16 consecutive address locations in your PC's I/O space. Appendix A provides an I/O port address map for your reference. This will assist you in locating an appropriate address for your peripheral devices and interface cards

I/O port base addresses are selected from the 6-position DIP switch, SW1, on-board the PCL-711B. Valid addresses are from 000 to 3F0 (hexadecimal). The factory default address setting is 220. From time to time, you may find that you will have to use some of these spaces for other devices. If this is the case, then you can change the address according to the information given in the following table

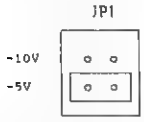
I/O ADDRESS RANGE (HEXADECIMAL)	SWITCH POSITION (SW1) 1 A9	2 A8	3 A7	4 A6	5 A5	6 A4
000 - 00F	0	0	0	0	0	0
100 - 10F	0	1	0	0	0	0
200 - 20F	1	0	0	0	0	0
210 - 21F	1	0	0	0	0	1
220 - 22F *	1	0	0	0	1	0
300 - 30F	1	1	0	0	0	0
3F0 - 3FF	1	1	1	1	1	1

NOTE: 0 = ON, 1 = OFF
A4 through A9 correspond to your PC's address lines.
* Factory default

2.2.2. D/A Range Selection

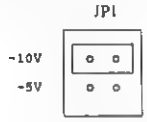
PCL-711B's D/A output range depends on the reference voltage you select at the jumper, JP1. The reference voltages can be assigned as either -5V or -10V, which gives you an D/A output range of 0 to +5V or 0 to +10V, respectively.

When an D/A output range of 0 to +5V is required, set the jumper to -5V (see the illustration below)



*Factory default

If your application requires an D/A output range of 0 to +10V, then set the jumper to -10V (refer to the illustration below).



NOTE: When -10V reference is selected, the maximum D/A output voltage should be +10V. However, it may be less than +10V because the +12V voltage source supplied by your PC may be lower than +11.5V which is required by the D/A circuit.

2.3. Connector Pin Assignment

The PCL-711B is equipped with three 20-pin connectors. Two connectors are located at CN3 and CN4. These are used for digital input (CN4), and digital output (CN3). The third connector is located at CN1, and is used for analog I/O.

Each of these connectors can be connected with the same type of ribbon cables. They can also be connected to a D37 connector using the PCLK-1050 industrial wiring kit.

An illustration of each of these connectors are given in the following illustrations.

Key:

A/D	=	Analog input
AGND	=	Analog ground
D/A	=	Analog output
D/O	=	Digital output
D/I	=	Digital input
DGND	=	Digital ground
VREF	=	Voltage reference

Analog I/O (CN1)

A/D 0	1	2	AGND
A/D 1	3	4	AGND
A/D 2	5	6	AGND
A/D 3	7	8	AGND
A/D 4	9	10	AGND
A/D 5	11	12	AGND
A/D 6	13	14	AGND
A/D 7	15	16	AGND
D/A	17	18	AGND
AGND	19	20	AGND

Digital Output (CN3)

D/O 0	1	2	D/O 1
D/O 2	3	4	D/O 3
D/O 4	5	6	D/O 5
D/O 6	7	8	D/O 7
D/O 8	9	10	D/O 9
D/O 10	11	12	D/O 11
D/O 12	13	14	D/O 13
D/O 14	15	16	D/O 15
DGND	17	18	DGND
+5V	19	20	+12V

Digital Input (CN4)

D/I 0	1	2	D/I 1
D/I 2	3	4	D/I 3
D/I 4	5	6	D/I 5
D/I 6	7	8	D/I 7
D/I 8	9	10	D/I 9
D/I 10	11	12	D/I 11
D/I 12	13	14	D/I 13
D/I 14	15	16	D/I 15
DGND	17	18	DGND
+5V	19	20	STROB

2.4. Plugging the PCL-711B into Your PC

Before you plug the PCL-711B into your PC, make sure that the computer's power is turned off, and that all power cords and peripheral devices have been disconnected from the system.

Use the following procedures as a guideline for plugging the PCL-711B into your computer.

1. Remove the cover from your PC's chassis, and locate a vacant expansion slot on your passive backplane or motherboard for installing the PCL-711B.
2. Take the card and insert its edge connector into the expansion slot, pressing the card firmly into place. Use the card's mounting bracket as a guide between the chassis' rear panel and backplane or motherboard.
3. Once you have inserted the card firmly into the slot, secure it to the chassis by fastening its mounting bracket with a screw.
4. Attach any ribbon cables to connectors CN1, CN3, and CN4 that your application may require.
5. Now, replace the chassis cover, and connect any power cords and peripheral cables that you disconnected.

The PCL-711B's installation is now complete.

2.5. Signal Connections

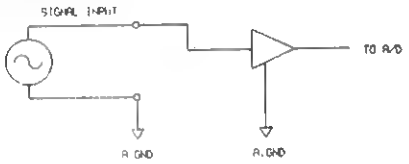
Since most data acquisition applications involve voltage measurement, it is important to make the correct signal connections in order to avoid any damage to your system, and insure accurate data acquisition. This chapter provides some helpful information about making the appropriate and proper signal connections for your application.

2.5.1. Analog Input Connection

As you already know, the PCL-711B supports eight single-ended analog inputs. A single-ended analog input connection uses only one signal wire connected to an analog input terminal which references to a common ground. For example, in order to measure a battery's voltage, simply connect its negative side to the PCL-711B's ground (any one of the AGND pins on connector CN1), and its positive side to one of the card's analog input channels

NOTE: The PCL-711B does not support differential signal source inputs.

The following diagram illustrates a single-ended, common ground, analog input connection.

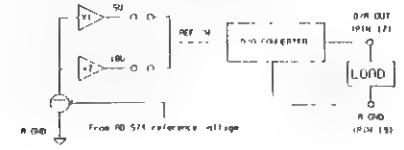


Single-Ended Analog Input Connection

2.5.2. Analog Output Connection

The PCL-711B is equipped with an internal -5 and -10V reference source which generates an output of 0 to +5 and 0 to +10V D/A output. The PCL-711B only provides one D/A output channel Use Connector CN1 for making your analog output

The following diagram illustrates a typical analog output or D/A connection:

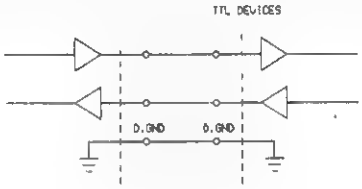


Analog Output Connection

2.5.3. Digital Signal Connection

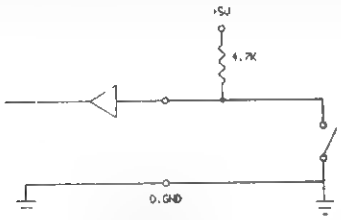
As mentioned in Chapter 1, the PCL-711B provides 16 digital input and 16 digital output channels for usage in digital input/output applications, such as timer/counter operations. All digital I/O levels are TTL compatible.

In order to transmit or receive a digital signal to or from other TTL devices, make your signal connection as illustrated in the following diagram:



Digital Signal Connection

To receive an OPEN or SHORT signal from a switch or relay , a pull up resistor must be installed on the PCL-711B. This resistor ensures that a high signal level will be maintained when the switch is open. Refer to the diagram illustrated below for an example of an OPEN/SHORT signal connection.



OPEN/SHORT Signal Connection

2.6. Software Driver

The PCL-711B is supported by Advantech's standard, user-friendly software driver. This driver allows you use standard functions, written in common programming languages, to operate the PCL-711B, without going into detailed register control. Languages supported by the software driver include BASICA, GWBASIC, QUICKBASIC, Microsoft C/C++ and PASCAL, Borland C/C++ and Turbo PASCAL Please refer to the manual for the software driver for more information.

CHAPTER 3. CONTROLLING THE PCL-711B

This chapter has been written for those of you who wish to write their own software driver instead of using the PCL-711B's. Here, you will find detailed information about the PCL-711B's register formats and control procedures.

3.1. I/O Port Address Map

The following table shows you which base I/O addresses are used by the PCL-711B. Refer to this map from time to time in order to become familiar with each of the card's register formats and their purpose. 16 consecutive registers corresponding to their I/O addresses are used to control the PCL-711B's various functions. The following table has been provided in this chapter as a preface which outlines these addresses relative to their location and control (read or write) assignments.

LOCATION	READ	WRITE
BASE+0	Counter 0	Counter 0
BASE+1	Counter 1	Counter 1
BASE+2	Counter 2	Counter 2
BASE+3	N/A	Counter Control
BASE+4	A/D low byte	D/A low byte
BASE+5	A/D high byte	D/A high byte
BASE+6	D/I low byte	N/A
BASE+7	D/I high byte	N/A
BASE+8	N/A	Clear interrupt status
BASE+9	N/A	Gain control
BASE+10	N/A	Multiplexer scan control
BASE+11	N/A	Mode and interrupt control
BASE+12	N/A	Software A/D trigger
BASE+13	N/A	D/O low byte
BASE+14	N/A	D/O high byte
BASE+15	N/A	N/A

The sections that follow provide further information about each register's data format according to its specific operation.

3.2. A/D Conversion

3.2.1. A/D Data Registers

The PCL-711B uses the data registers located at I/O ports BASE+4 and BASE+5 to store the converted A/D data. The low byte data is stored at BASE+4, and the high byte data is stored at BASE+5

BASE+4 A/D Low Byte Data (Read)

D7	D6	D5	D4	D3	D2	D1	D0
AD7	AD6	AD5	AD4	AD3	AD2	AD1	AD0

BASE+5 A/D High Byte Data (Read)

D7	D6	D5	D4	D3	D2	D1	D0
0	0	0	DRDY	AD11	AD10	AD9	AD8

Where:

AD0 through AD11: Represent the PCL-711B's A/D data bits. AD0 is the Least Significant Bit (LSB), and AD11 is the Most Significant Bit (MSB).

DRDY: Data ready bit. When A/D conversion is in progress, this bit remains as 1. It becomes 0 when the A/D conversion is completed. It will become to 1 after reading the low byte A/D data from BASE+4.

3.2.2. Gain Control Register

BASE+9 is used to set the PCL-711B's amplification gain for A/D conversion. The PCL-711B provides five different gains: x1, x2, x4, x8, and x16.

The following tables outline BASE+9's register format and corresponding gain settings:

BASE+9 Gain Control Register (Write)

D7	D6	D5	D4	D3	D2	D1	D0
-	-	-	-	-	G2	G1	G0

G2	G1	G0	GAIN
0	0	0	x1
0	0	1	x2
0	1	0	x4
0	1	1	x8
1	0	0	x16

3.2.3. Multiplexer Scan Register

The PCL-711B can multiplex up to 8 channels of analog input. Users have to set this register, located at BASE+10, to select to the desired channel , which is going to be measured, before performing any A/D conversion. The register format is as below:

BASE+10 Multiplexer Scan Control (Write)

D7	D6	D5	D4	D3	D2	D1	D0
-	-	-	-	-	C2	C1	C0

C2	C1	C0	CH.
0	0	0	0
0	0	1	1
0	1	0	2
0	1	1	3
1	0	0	4
1	0	1	5
1	1	0	6
1	1	1	7

3.2.4. Mode and Interrupt Control Register

The PCL-711B's A/D conversion can be triggered in one of the following three ways:

- **By software**
  Writing any data to BASE+12 will generate a trigger pulse to the PCL-711B's on-board A/D converter.
- **By the on-board clock (pacer)**
  The PCL-711B is equipped with a Intel 8253, a programmable interval timer/ counter, to generate precise clock output (pacer). The pacer clock rate of the PCL-711B is between 0.5MHz and 35 minutes per pulse. (See section 3.7 for details on programming the Intel 8253 timer/counter.)
- **By external pulse**
  The PCL-711B allows users to use external signal, from D/I 0 (Pin 1 of the connector CN4), as the A/D trigger pulse. The A/D conversion will be triggered at the rising edge of the external signal.

Also, the PCL-711B provides two ways to transfer the converted A/D data to certain variables:

- **By software control (foreground)**
  The software control data transfer utilizes advantage of the foreground polling concept. After the A/D converter has been triggered, the application program should keep checking the DRDY bit of I/O port BASE+5 until the DRDY bit is detected as 0. Then the program should read data from BASE+4 and BASE+5 to get the whole converted data.
- **By interrupt (background)**
  The PCL-711B also provides background data transfer support, if it is programmed to be in the interrupt data transfer mode. If the PCL-711B is in interrupt data transfer mode, it will generate an interrupt to your PC after each A/D conversion is completed. The corresponding ISR (interrupt service routine) should handle everything to transfer the converted data to memory variables in your program.

I/O port BASE+11 is used to set the PCL-711B's operation mode and the IRQ level. The register format is as following:

BASE+11 Mode and Interrupt Control Register (Write)

D7	D6	D5	D4	D3	D2	D1	D0
-	I3	I1	I0	-	S2	S1	S0

Where

S0 to S2: mode selection

S2	S1	S0	Operation Mode
0	0	0	S/W trigger with S/W data transfer
0	0	1	
0	1	0	External trigger * with S/W data transfer
0	1	1	External trigger * with INT data transfer
1	0	0	Pacer trigger with S/W data transfer
1	0	1	Reserved
1	1	0	Pacer trigger with INT data transfer
1	1	1	Reserved

Note: External trigger signal go through DI0 of CN4

I0 to I2: IRQ level selection

I2	I1	I0	Interrupt Level
0	0	0	IRQ2
0	0	1	N/A
0	1	0	IRQ3
0	1	1	IRQ3
1	0	0	IRQ4
1	0	1	IRQ5
1	1	0	IRQ6
1	1	1	IRQ7

3.2.5. Interrupt Status Register

If the PCL-711B is in interrupt data transfer mode, a hardware status flag will be set after each A/D conversion. Users have to clear the status flag, by writing any data to BASE+8, to let the PCL-711B accept next interrupt.

BASE+8 Clear Interrupt Status (Write)

D7	D6	D5	D4	D3	D2	D1	D0
-	-	-	-	-	-	-	-

3.2.6. Software Trigger Register

Writing any data to BASE+12 will generate a trigger pulse to the PCL-711B's A/D converter.

BASE+12 Software A/D Trigger (Write)

D7	D6	D5	D4	D3	D2	D1	D0
-	-	-	-	-	-	-	-

3.3. D/A Conversion

The PCL-711B provides one D/A output channel. The low byte and the high byte D/A data are set via BASE+4 and BASE+5 respectively.

Since the PCL-711B uses so-called double-buffer D/A output technology to avoid output glitch, the low byte data should be written first and the high byte data second, that is write BASE+4 first and BASE+5 second. The D/A output will not change until BASE+5 is updated.

BASE+4 D/A Low Byte Data (Write)

D7	D6	D5	D4	D3	D2	D1	D0
DA7	DA6	DA5	DA4	DA3	DA2	DA1	DA0

BASE+5 D/A High Byte Data (Write)

D7	D6	D5	D4	D3	D2	D1	D0
-	-	-	-	DA11	DA10	DA9	DA8

where:

DA0 through DA11: the D/A output data. DA0 represents the D/A's LSB data, while DA11 represents the D/A's MSB data.

3.4. Digital Input and Output

3.4.1. Digital Input Registers

The PCL-711B provides 16 bits of digital input. The registers are located at BASE+6 and BASE+7.

BASE+6 D/I Low Byte Data (Read)

D7	D6	D5	D4	D3	D2	D1	D0
DI7	DI6	DI5	DI4	DI3	DI2	DI1	DI0

BASE+7 D/I High Byte Data (Read)

D7	D6	D5	D4	D3	D2	D1	D0
DI15	DI14	DI13	DI12	DI11	DI10	DI9	DI8

3.4.2. Digital Output Registers

The PCL-711B provides 16 bits of digital output. The registers are located at BASE+13 and BASE+14.

BASE+13 D/O Low Byte Data (Write)

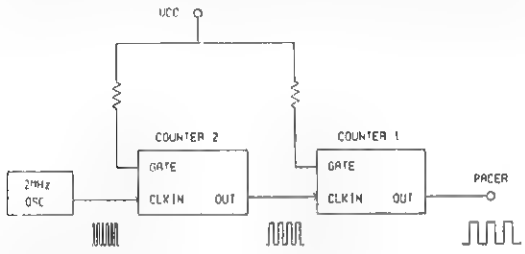
D7	D6	D5	D4	D3	D2	D1	D0
DO7	DO6	DO5	DO4	DO3	DO2	DO1	DO0

BASE+14 D/O High Byte Data (Write)

D7	D6	D5	D4	D3	D2	D1	D0
DO15	DO14	DO13	DO12	DO11	DO10	DO9	DO8

3.5. Pacer Programming

The PCL-711B uses an Intel 8253, a 16-bit programmable counter/timer, to generate pacer clock. Each Intel 8253 provides three independent counter/timer channels, Counter 0, Counter 1 and Counter 2. The PCL-711B cascades Counter 1 and Counter 2 as a 32-bit frequency divider to support wide range of pacer clock rate, as shown below



Intel 8253 has six operation modes, from Mode 0 through Mode 5. To generate pacer clock, both Counter 1 and Counter 2 should be programmed as Mode 2 (rate generator mode).

Four I/O ports, from BASE+0 through BASE+3, are used to program the on-board Intel 8253:

BASE+0: Counter 0 (Read/Write)
BASE+1: Counter 1 (Read/Write)
BASE+2: Counter 2 (Read/Write)
BASE+3: Counter Control (Write only)

Please refer to the following steps to set desired pacer clock rate

Step 1: Write '74H' to BASE+3 to make Counter 1 work at Mode 2

Step 2: Write an appropriate data (16-bit data, ranging from 2 to 65535) to BASE+1 to set Counter 1's divisor constant C1. Since C1 is a 16-bit data, you have to first write the low byte of C1 to BASE+1, then write the high byte of C1 to BASE+1

Step 3: Write 'B4H' to BASE+3 to make Counter 2 work at Mode 2.

Step 4: Write an appropriate data (16-bit data, ranging from 2 to 65535) to BASE+1 to set Counter 1's divisor constant C2. Since C2 is a 16-bit data, you have to first write the low byte of C2 to BASE+2, then write the high byte of C2 to BASE+2.

The pacer rate is determined by the following formula:

Pacer rate = (2 MHz)/(C1*C2)

In the following example (written in BASIC), C1 is set as 40 and C2 is set as 10, thus the pacer rate will be 5 KHz (5KHz=2MHz/(40*10)).

```
500  OUT (BASE+3,&H74)     ' Set Counter 1 as Mode 2
510  OUT (BASE+1,40)       ' write low byte of C1
520  OUT (BASE+1,0)        ' write high byte of C1
530  OUT (BASE+3,&HB4)     ' Set Counter 2 as Mode 2
540  OUT (BASE+2,10)       ' write low byte of C2
550  OUT (BASE+2,0)        ' write high byte of C2
```

NOTE 1: Counter 0 is reserved to future dèvelopement.

NOTE 2: For more detailed information about Intel 8253's register formats, please refer to Appendix B.

APPENDIX A. CALIBRATION

In data acquisition and control application, it is important to constantly calibrate your measurement device to maintain its accuracy. A calibration program, CAL711B.EXE is provided in the PCL-711B software disk to assist your calibration work.

Minimum equipment required to perform a satisfactory calibration is a 4 1/2 digit digital multimeter. In addition, a voltage calibrator or a stable noise free d.c. voltage source that can be used in conjunction with the digital multimeter is required. A card extender, such as the PC-LabCard model PCL-757 is an inexpensive device that you will find greatly improves access to the board during calibration and will probably be useful with other boards.

Calibration is easily performed using the CAL711B.EXE program. This program will lead you through the calibration and set up procedure with variety of prompts and graphic displays directing you to correct settings and adjustment the variable resistors. The explanatory material in this section is brief and is intended for use in conjunction with the calibration program.

A.1. VR Assignments

The PCL-711B has five on-board VRs, which allow you to make accurate calibration adjustments for the card's A/D and D/A functions. Each VR's location is indicated in Figure A-1. The function of each VR is listed below:

VR1 D/A full scale adjustment
VR2 D/A offset adjustment
VR3 A/D offset adjustment
VR4 A/D full scale adjustment
VR5 Programmable amplifier offset adjustment

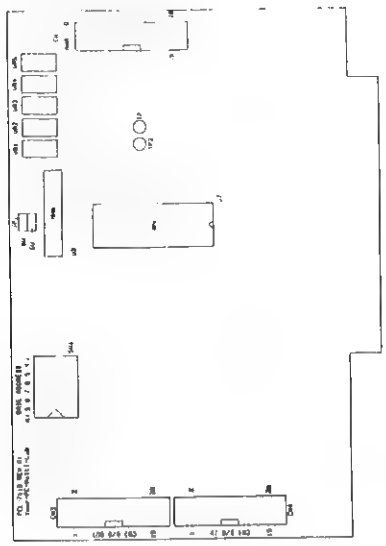


Figure A-1 VR location

A.2. D/A Calibration

The user should first choose the D/A output range to be calibrated by setting the J1 (5V or 10V). The zero offset and full scale of D/A channel can be tuned through two VRs: VR1 is for the full scale adjustment of D/A and VR2 is for the zero offset adjustment of D/A. The user should use a precision voltmeter to measure the D/A output.

Calibration steps:

1. Full scale adjustment. The D/A digital data is sent to 4095. Adjust VR1 until the reading of your voltmeter equals to VREF(reference voltage of D/A output) with opposite sign.
2. Offset adjustment. The D/A digital data is sent to 0. Adjust VR2 until the reading of your voltmeter is 0 volts

A.3. A/D Calibration

Because the PCL-711B supports versatile A/D input ranges, accurately calibrated result for certain A/D range may still cause a small offset for the other ranges. It is suggested that you make a calibration again when you are going to use different A/D input range.

Calibration steps:

1. Short the A/D input of Channel 0 to AGND. Then, adjust the VR3 until the reading of the A/D conversion flickers between 2047 and 2048.
2. Apply a voltage with the full sacle value corresponding to the specific A/D input range to A/D Channel 0. Then,adjust the VR4 until the reading of A/D conversion flickers between 4094 and 4095.

APPENDIX B. INTEL 8253 REFERENCE

B.1. Operation Modes

B.1.1. Mode 0 Stop on Terminal Count

In Mode 0, the counter/timer's output will initially be set low. The output will remain low, and the counter will start to count after the count has been loaded into the selected count register. When the terminal count has been reached, the output will be set high, and remain high until the selected counter is reloaded with this mode or a new count has been loaded. The counter will continue to decrement after the terminal count has been reached. Rewriting data to a counter register during counting generates the following results:

1. Writing to the first byte stops the current count.
2. Writing to the second byte starts a new count.

B.1.2. Mode 1 Programmable One-Shot

When in this mode, the output will be set low on the count that follows the gate input's rising edge. The output goes high on the terminal count. If a new count value is loaded while the output is set low, then it will not affect the duration of the one-shot pulse until after the succeeding trigger. The current count can be read at any time without affecting the one-shot pulse. The one-shot is retriggerable. This allows the output to remain low for the full count after any rising edge from the gate input.

B.1.3. Mode 2 Rate Generator

When in Mode 2, the counter/timer's output will be set at low for one period of the input clock. The period from one output pulse to the next is equal to the number of input counts in the counter register. If the counter register is reloaded between output pulses, the present period will not be affected.

B 1.4. Mode 3 Square Wave Generator

This mode is similar to Mode 2, with the exception that the output remains high until one half of the count value has been completed (for even values), and then low for the other half of the count. This is accomplished by decreasing the counter by two on the falling edge of each clock pulse. When the counter reaches the terminal count, the output's state will be changed. The counter will be reloaded with the full count, repeating the entire process.

If the count value is odd, and the output high, then the first clock pulse (after the count is loaded) decrements the count by one. Subsequent clock pulses will decrement the count by two. After a timeout period, the output is set low, and the full count value is then reloaded. The clock pulse (following the reload) decrements the counter by three. Subsequent clock pulses decrement the count by two until a timeout occurs, then the whole process is repeated. In this way, if the count value is odd, the output will be set high for (N + 1)/2 counts, and low for (N - 1)/2 counts.

B.1.5. Mode 4 Software Triggered Strobe

Mode 4 sets the counter/timer's output high. When the count value is loaded, the counter begins counting. The output will be set low for one input clock period when the terminal count is reached, after which it will be set high again.

If the count register is reloaded during counting, the new count will be loaded on the next clock pulse. Also, the count will be inhibited while the gate input is low.

B.1.6. Mode 5 Hardware Triggered Strobe

When in Mode 5, the counter will start counting after the trigger input's rising edge, and will then be set low for one clock period when the terminal count is reached. The counter is retriggerable in this mode.

B.2. Counter Control Register Format

BASE + 3 Counter Control Register (Write)

D7	D6	D5	D4	D3	D2	D1	D0
SC1	SC0	RW1	RW0	M2	M1	M0	BCD

Key:

SC1 & SC0 Select Counter

SC1	SC0	COUNTER
0	0	0
0	1	1
0	0	2
1	1	illegal

RW1 & RW0 Select Read/Write Operation

RW1	RW0	OPERATION
0	0	Counter latch
0	0	Read/Write LSB
0	1	Read/Write MSB
1	1	Read/Write LSB first, then MSB

M2, M1, & M0 Select Operation Mode

M2	M1	M0	MODE
0	0	0	0 = Interrupt on terminal count
0	0	1	1 = Programmable one shot
X	1	0	2 = Rate generator
X	1	1	3 = Square wave rate generator
1	0	0	4 = Software triggered strobe
1	0	1	5 = Hardware triggered strobe

BCD Select Binary or BCD Counting

BCD	PURPOSE
0	16-bit binary counter
1	Binary coded decimal (BCD) counter with four decades

The BCD is defaulted to count in binary mode. The count can be set to any value from 0 up to 65535. If you set this bit to BCD (logic 1), then the count may be set to any value from 0 up to 9999.

ADVANTECH

PC-LabCard Software Drivers

TABLE OF CONTENTS

CHAPTER 1. INTRODUCTION

1.1. Basic Concepts

The Advantech Data Acquisition PC-LabCards come with their own integrated software driver which simplifies programming. All the drivers support the same convention.
This way software is isolated from hardware, and consistent application programs can be created.

The general capabilities supported are:

- **A/D Conversion** Performs analog to digital conversion.
- **D/A Conversion** Performs digital to analog conversion.
- **Digital Input** Inputs a logical(1 or 0) signal.
- **Digital Output** Outputs a logical(1 or 0) signal.
- **Counter/Timer** Supports frequency measurement, event counting, pulse output, timer interval measurement, and timed interrupt generation.
- **Daughter Board** Supports functions for utilizing the daughter board to enhance PC-LabCard capabilities.
- **Block Scan** Performs A/D conversions almost concurrently on a block of input channels at constant time interval.

Although this standard specification offers a comprehensive set of functions for analog and digital I/O, as well as for counter/timer functions, not every driver supports the entire set of functions. Most PC-LabCards do not offer all the functionality in their hardware needed to support all the functions.

Each PC-LabCard has its own driver. It is recommended that you use the driver to develop your application. Programming languages supported by the driver include BASICA, GWBASIC, Quick BASIC 4.0/4.5, Microsoft C, Turbo C, Borland C++, Microsoft PASCAL, and Turbo PASCAL. The driver is a **Terminate and Stay Resident (TSR)** program which runs in the background while your application runs in the foreground. Before running your application

program, you must load the driver into your system's memory To do this, enter driver name at the DOS prompt from your keyboard (for example, type "PCL-718" at DOS prompt for the PCL-718)

Each driver supports up to two of the same type PC-LabCards at one time For example, driver PCL-718 supports two PCL-718 cards at one time. If you want to use two (or more) different types of PC LabCards at one time, you need to load the drivers for each different cards Drivers are prevented from being repeatedly loaded into the PC's memory To release the driver from your PC's main memory, execute the Free exe utility

1.2. Parameter Table

In the past, with many older software drivers it was necessary to issue a long list of function calls to perform a simple A/D conversion. For example, your programming procedure might look something like this:

Step 1. Use FUN 1 to initialize the hardware.
Step 2. Use FUN 2 to set the input range.
Step 3. Use FUN 3 to set start and stop scanning channels.
Step 4. Use FUN 4 to set the A/D trigger mode.
Step 5. Use FUN 5 to set the pacer clock rate.
Step 6. Use FUN 6 to set the IRQ level.
Step 7. Use FUN 7 to set the DMA channel.
Step 8. Use FUN 8 to perform A/D conversion 'N' times.
Step 9. Use FUN 9 to check the operation status.

As you can imagine, performing these procedures each time you write your program becomes very time consuming and frustrating. Why should you have to remember so many function calls and their corresponding parameters?

The software driver simplifies your programming by using a Parameter Table reference algorithm. The tables hold parameters, masks, minimum and maximum values, and other specific information regarding the functions. In contrast, the application program contains tables specifying parameters and modes of operations. All the function calls supported by the drivers need only

two arguments, the function number and a memory address pointer which points to a pre defined Parameter Table.

Once the Parameter Table is defined, just assign the desired function number and the Parameter Table's address to the driver Once this is done, it will pick up the necessary parameters associated to that specific function call, and then automatically execute the function.

A Parameter Table's format will be illustrated in detail later in this manual. What you have to know here is that the Parameter Table includes all the parameter settings necessary for all data acquisition function calls supported by the software driver, such as A/D start and stop channel numbers, number of A/D conversion, pacer rate setting, etc. Following are some examples of parameter tables:

```
-------------------------------------------------------------------
* Example for C:

extern pcl718(int, unsigned int *);
unsigned param[60];       /* Parameter Table */
unsigned buffer[1000];    /* A/D data buffer */

main()
{
  unsigned far *ptr;      /* buffer pointer */

  ptr = (unsigned far *) buffer;
  param[0] = 0;           /* card number              */
  param[1] = 0x200;       /* I/O base address         */
  param[4] = 2;           /* IRQ level 2              */
  param[10] = FP_OFF(ptr);/* offset address           */
  param[11] = FP_SEG(ptr);/* segment address          */
  param[12] = 0;          /* only one buffer used     */
  param[13] = 0;
  param[14] = 1000;       /* number of A/D conversion */
  param[15] = 0x0;        /* A/D start channel        */
  param[16] = 0xA;        /* A/D stop  channel        */
  param[17] = 0x0;        /* gain code                */

  pcl718(3,param);        /* initialize the PC-LabCard */
  pcl718(4,param);        /* initialize A/D function   */
  pcl718(5,param);        /* A/D conversions, and store*/
                          /* converted data to buffer  */
```



```
-------------------------------------------------------------------
* Example for BASIC:
        .
100 DIM param%(60)'Parameter Table
    .
    .
    .
320 func_no% = 3
330 CALL pcl718(fun_no%, param%(0))
```

```
-------------------------------------------------------------------
* Example for PASCAL:

var
param : array[0..60] of word;{ Parameter Table }
func  : integer;
procedure pcl718(func:integer; var param:word); external;
    .
    .
    .
func := 3;
pcl718(func, param[0]);
```

```
-------------------------------------------------------------------
```

The Parameter Table reference algorithm is more sophisticated, powerful, and easier to use than older types of drivers. There are two ways to change your program's settings:

1. Modify corresponding parameters directly.
   If you want to change the A/D start channel number, for example, you do not need to issue any function calls to change this setting, just change the corresponding parameter in your Parameter Table.

   Example (C language):

```
extern pcl718(int, unsigned int *);
unsigned int param[60];/* Parameter Table*/

main()
{
.
.
.
```



```
param[15] = 0x0;      /* A/D start channel            */
param[16] = 0xA;      /* A/D stop  channel            */
pcl718(5,param);      /* S/W triggered A/D conversion */
.
.
param[15] = 0x2;      /* A/D start channel            */
pcl718(5,param);      /* S/W triggered A/D conversion */
.
.
}
```

2. Create a new Parameter Table

   The software driver's job oriented algorithm gives your program the capability of addressing several Parameter Tables using the same function call or group of function calls in one program. It should be noted, however, that the driver can only address one Parameter Table at a time. The driver executes the jobs according to the specified Parameter Table. For programming conveniently, we can previously define individual Parameter Table for each of frequently called functions without troublesome table modification.

   Example(C language):

```
extern pcl718(int, unsigned int *);
unsigned param1[60];/* Parameter Table 1          */
unsigned param2[60];/* Parameter Table 2          */

main()
{
.
.
.
/* JOB 1 */
param1[15] = 0x0; /* A/D start channel            */
param1[16] = 0xA; /* A/D stop  channel            */
param1[17] = 0x0; /* gain code                    */
pcl718(5,param1); /* S/W triggered A/D conversion */
.
.
.
/* JOB 2 */
param2[15] = 0x2; /* A/D start channel            */
param2[16] = 0x8; /* A/D stop  channel            */
param2[17] = 0x5; /* gain code                    */
pcl718(5,param2); /* S/W triggered A/D conversion */
.
.
.
}
```

Two Parameter Tables are defined in this example. Job 1, and 2 are the same, except for the start channel, stop channel and gain setting

NOTE: When using the BASIC language, negative numbers must be used to represent integer data over 32767

negative number = integer - 65536

For example, you need to pass 45000 as a input parameter to BASIC function. 45000 - 65536 = -20536 So, you have to use -20536 rather than 45000 as a parameter for the BASIC function.

1.3. Data Buffer

While writing your application programs using the driver, a data buffer is commonly used for most function calls. That is, the driver usually reads or writes data from/to a data buffer instead of variables.

A data buffer is nothing more than a block of consecutive memory. It can be in any form which your programming language supports. For example, you can declare and reserve the memory space for an integer array in advance or allocate a block of memory space dynamically as needed.The size of data buffer can be anywhere below 1 megabyte (in PC real mode), from two bytes up to 64K bytes, depending on your requirements.

No matter what the size of your data buffer is, before you perform any function calls, you must pass its segment and offset addresses (or the memory pointer which points to the specific data buffer) to the corresponding parameters in your **Parameter Table.**

Although the size of buffer can be up to 64K, **you should avoid possibility of overflow of the offset address.** For example, offset address is 8000(hex) and buffer size is A000(hex). 8000h + A000h = 12000h, this overflow makes address pointing to wrong location. You can fix the problem either to cut buffer size or to allocate another buffer so that its offset is correct.

The software driver allows you to define and use one or two memory buffers. The first buffer is designated Buffer A, the second is designated **Buffer B.**

Data returned from an A/D conversion will occupy two bytes of space Data for a D/A conversion will also occupy two bytes of space (In each case, the first will be a low byte, the second will be a high byte) Digital I/O occupies only one byte for each conversion. Each time you perform an A/D, D/A or digital conversion, the driver stores/retrieves the data to/from the memory buffer, filling/getting it in sequential order, until all conversions have been completed.

It should be noted that the buffer size should not be less than the number of bytes on which you intend to perform an A/D, a D/A or a digital conversion operation.

If you use two buffers simultaneously, you will need to define two data buffers with the same buffer size. In the case of an A/D conversion function, the driver will store data N times in Buffer A first then N times in Buffer B. N is specified in the parameter table. The D/A and digital conversions work in the same way.

NOTE: Double buffers are not supported in **software data transfer** mode.

1.4. Cyclic Operation

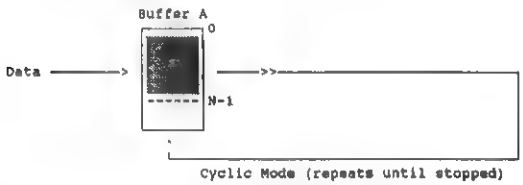
The software driver provides two types of operation modes: Cyclic and Non-cyclic. As you will see, when an A/D conversion is being performed, the acquired data will be stored to the data buffer that you defined, sequentially filling its space as each conversion is made. For example, you might want to perform A/D conversions N times. If you are using the Non-cyclic Mode of storing your data, the driver will begin storing data in sequential order beginning with conversion number 1 until it reaches the Nth conversion. If you are performing A/D conversions while in Cyclic Mode, then data will continue to be sequentially stored in the same buffer from the beginning (the first space) of that buffer, and repeats the sequence until you issue a **STOP** command.

The software driver's Cyclic Mode can also be used with the double-buffered feature. If you define two buffers, A and B, the driver will first store data in Buffer A N times, then in Buffer B N times. After data has been stored in Buffer B, the process repeats, filling Buffers A and B in sequence, until a **STOP** command is issued. This feature allows you to simulate "near" real-time data acquisition applications. For example, you can acquire data in the background, and simultaneously display other data in the foreground. However,

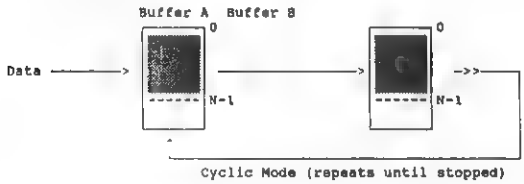
this method does have its drawback. Since it takes time to switch between buffers (depending on your CPU's performance), **some of your data may become lost during the switching period.** Needless to say, you should use this method of data storage with caution.

The D/A and Digital I/O conversions also work the same way as described above.

- **Single Buffer Mode:**



- **Double Buffer Mode:**



1.5. Trigger Mode

Each A/D (analog to digital) conversion, D/A (digital to analog) conversion or (D/O) digital output function must be operated under internal clock trigger (pacer trigger) mode or external clock trigger mode.

The D/I digital input function can only be triggered by internal clock.

1. **Internal Clock Trigger (or Pacer Trigger) Mode**
   This mode uses the PC-LabCard's internal clock as a trigger source for the conversion operation. The internal clock is generated from an on-board Timer/Counter chip (such as the 8253 or 9513). The desired clock rate can be precisely and easily programmed, as described in detail in a later section.

2. **External Clock Trigger Mode**
   In some applications, you may want to trigger conversions when an external signal changes. The operation is similar to the pacer trigger except that an external clock generates the conversion trigger. Please refer to your hardware manual for the pin connections for the external clock, jumper settings for this mode, and for clock rate limitations.

1.6. Analog to Digital Conversion

1.6.1. Gain Array Table

When performing A/D conversions, you must set each channel's amplification gain to a specific value. For the highest possible resolution, the signal should be amplified so that the maximum voltage swing equals the maximum input range of the A/D convertor. The driver provides two alternatives to help you set the gain for each channel.

First, you can define a Gain Table which consists of consecutive words of memory. The elements number in this array must be as same as the channel number on hardware even though you do not use that many of the channels. Each word corresponds to the amplification gain code for each of the analog input channels. Then pass the offset and segment addresses of the Gain Table to param[18] and param[19] respectively. This feature allows you to define different Gain Array Tables conveniently for whatever A/D conversion application you wish to perform.

The number of analog input channel depends on hardware. For example, 16 channels are supported by PCL-814 but only 8 channels are supported by the PCL-711B. Please refer to your individual hardware manual for the gain code for each input amplification gain.

Gain Table

Word 0	Channel 0's Code
Word 1	Channel 1's Code
Word 2	Channel 2's Code
	.
.	.
	.
	.
Word 14	Channel 14's Code
Word 15	Channel 15's Code

Second, if your amplification gains for each channel are identical, then simply set the Parameter Table's param[17] to the gain code you will be using within your application. Otherwise, set param[17] to FF(hex) to cause the driver to refer the gain array table.

Example:

PCL-711B Gain Code Table

Input Range	Recommended Gain	Gain Code
+/- 5V	x1	0
+/- 2.5V	x2	1
+/- 1.25V	x4	2
+/- 0.625V	x8	3
+/- 0.3125V	x16	4

PCL-812PG Gain Code Table (JP9 set to +/- 5V)

Input Range	Recommended Gain	Gain Code
+/- 5V	x1	0
+/- 2.5V	x2	1
+/- 1.25V	x4	2
+/- 0.625V	x8	3
+/- 0.3125V	x16	4

PCL-812PG Gain Code Table (JP9 set to +/- 10V)

Input Range	Recommended Gain	Gain Code
+/- 10V	x1	0
+/- 5V	x2	1
+/- 2.5V	x4	2
+/- 1.25V	x8	3
+/- 0.625V	x16	4

PCL-818 Gain Code Table

Input Range	Recommended Gain	Gain Code
+/- 10V	x0.5	8
+/- 5V	x1	0
+/- 2.5V	x2	1
+/- 1.0V	x5	2
+/- 0.5V	x10	3
0 to 10V	x1	4
0 to 5V	x2	5
0 to 2V	x5	6
0 to 1V	x10	7

PCL-814 Gain Code Table

Input Range	Recommended Gain	Gain Code
+/- 5V	x1	0
+/- 2.5V	x2	1
+/- 1.25V	x4	2
+/- 0.625V	x8	3
0 to 10V	x1	4
0 to 5V	x2	5
0 to 2.5V	x4	6
0 to 1.25V	x8	7

1.6.2. A/D Start Channel and Stop Channel

The driver will automatically and sequentially scan each channel's analog input according to the start and stop channel number designated in your program.

For example, if the start channel number is 3, and the stop channel number is A(hex), then the scanning sequence will be automatically performed by channels 3,4,5,6,7,8,9,A,3,...., repeating the sequence.

1.6.3. A/D Data Transfer

The driver supports three different modes to perform data transfer (depending on hardware support). They are:

1. Software Data Transfer

This is the simplest way to do the A/D conversion. The function performs A/D conversions N times. For the time interval between conversions to be equal, the driver can utilize an on-board internal clock (pacer) for a timing signal. The driver triggers an A/D conversion and waits for the next clock pulse to do the next conversion. The process continues until the Nth conversion is completed. The driver returns to the application only when the Nth conversion is completed. If N is 1, only one A/D conversion is performed. The driver makes the conversion and returns to application immediately without waiting for the internal/external triggering signal. Double-buffer and Cyclic Mode are not supported by the Software Data Transfer mode.

2. Interrupt Data Transfer

The driver returns control to the application immediately. The A/D conversions are processed in the background and when one conversion is done, the driver will be interrupted and will then save the A/D data. This process is continued until the Nth conversion is completed or a STOP function is issued. Not all PC-LabCards support this mode.

3. DMA data transfer

The driver returns control to the application immediately. The A/D conversions are processed in the background and when one conversion is done, the driver will be interrupted and will then activate the DMA to transfer the A/D data. This process is continued until Nth conversion is completed or a STOP function is issued.

Not all PC-LabCards support this mode

This driver supports dual-DMA capability. That is, one DMA channel is used to save buffer A and another DMA channel is used to save buffer B, so data loss due to buffer changing will be minimized. Dual-DMA is not yet supported by hardware (This feature has been reserved for future hardware development), so both DMA channels must be set to the same number when using this mode.

1.6.4. A/D Data Buffer

In order to store data that is acquired from an A/D conversion, a data buffer must be created. This buffer consists of a block of consecutive system memory that must first be defined. The memory block must not have any other program occupying its space

1.6.5. Block Channel Scan

The block channel scan function allows the user to perform A/D conversions almost concurrently on a block of input channels, at a constant time interval which is set by the pacer.

On each pacer trigger pulse, the driver performs software triggers on all the input channels of the block and save data into memory. The block of input channels is specified by "A/D start channel param[15] " and "A/D stop channel param[16]".

For example, connect PCLD-787 with PCL-812, PCL-812PG, PCL-718 or PCL-818 to do simultaneous data acquisition. For more information about PCLD-787, please refer to PCLD-787 User's Manual.

1.7. Digital to Analog Conversion

1.7.1. D/A Start Channel and Stop Channel

The software driver gives you the advantage of sequentially performing multi-channel D/A conversions. Before performing a conversion, you will need to specify which channel numbers will start and stop the operation

1.7.2. D/A Data Buffer

To perform a D/A output operation, the driver will read digital data pre-stored in a user-defined memory buffer. To do this, a block of consecutive system memory must be defined. The memory buffer can contain up to 65536 bytes or 32768 words. (Each data element will occupy two bytes· the first one is the low byte, and the second is the high byte).

1.8. Digital Input/Output

The PC-LabCard series is equipped with its own on-board digital I/O, generally 2 ports for digital I/O (each port has 8 channels). Some hardware devices are equipped with a digital I/O module that can have up to 8 ports for digital I/O.

1.9. Counter/Timer

There are five counter/timer functions supported. The driver may not support all the functions depending on the hardware implementation.

1.9.1. Timer Interrupt

This function provides a way for application programs to do something periodically through hardware timer interrupts.

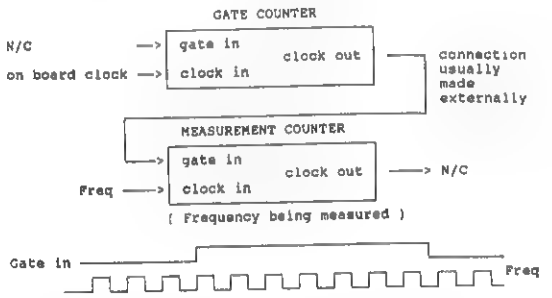
This assumes the availability of a timer on the PC-LabCard that is capable of generating a hardware interrupt on a periodic basis. The application program must provide the routine for periodic execution and set its address to INT 64h before enabling the timer interrupt function. The driver will make a long call to the address specified at INT 64h on each timer interrupt. (Not to make an interrupt, just to use the interrupt vector table as a location to store the address.)

This interrupt has many possible uses for an application program. For example, a program may use this periodic interrupt to perform a quick A/D operation, then a quick D/A operation, and then a plot of the A/D data.

1.9.2. Frequency Measurement

Frequency measurement requires the use of two timer channels. One is used as a "gate counter" and the other is used as a "frequency counter". The gate counter defines a gate time during which pulses are counted on the measurement counter

The user will need to supply the connection between the clock output of the"gate counter" and the gate input of the "measurement counter". The "gate counter" is implicitly fixed at timer channel 0 (the first one).



The driver is only responsible for returning the "measured count" value. The actual frequency must be calculated by the application program.

Frequency = (measured count)/((timer tick duration)*(gate count))

For example:

time tick duration = 0 5 micro second.
(2 Megahertz crystal on board).
gate count = 10000
measured count = 100

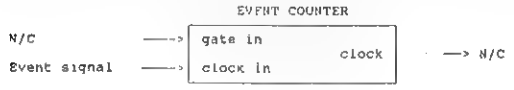
frequency = 100 / (0.5us * 10000) = 2 KHz.

1.9.3. Event Counting

For these functions, an external, non periodic signal is attached to the clock input of the event counter channel specified in the table

The Event Counting process is started via a start function call then the initial count value is set to 0 and clock input increases the count value Count value can be read via Event Count Read function Finally, the process is stopped via a stop function call and the final event count is returned.

For the PCL-718,PCL 818,PCL-812,and PCL-812PG, only channel 0 can be selected as the Event Counter

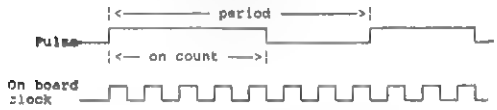


1.9.4. Pulse Output

These functions are used to generate an output signal with the "period" and "on count" specified in the table.

Period: Ticks of on board clock for a pulse high and low period.

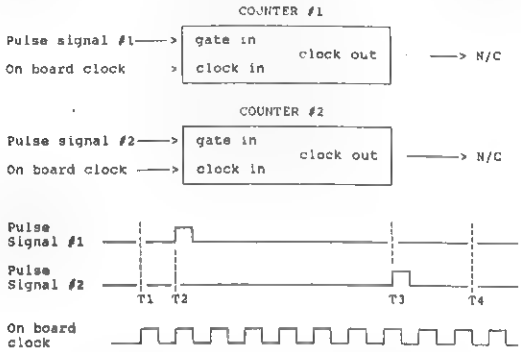
On Count: Ticks of on board clock for a pulse high period.



1.9.5. Time Interval Measurement

These functions are used to measure the time interval between two pulses It may be implemented by using two edge-triggered counter channels Because of the edge triggered constraint, **9513 but not 8253 based systems can implement this function** The pulses are connected to timer channels which are specified in the table. In order to measure the time interval between two pulses on the same line, the two input channels in the table are set equal to each other

The actual measurement process can be understood from the following timing diagram.



T1: Timing interval process is started via function call; neither counter is counting because neither has been triggered yet.

T2: Pulse Pin #1's rising edge triggers COUNTER #1 that begins to count at a rate specified by the on-board clock; COUNTER #2 has not yet begun to count because it has not yet been triggered.

T3: Pulse Pin #2's rise finally triggers COUNTER #2 which begins to count at the same rate as COUNTER #1, which is continuing to count.

T4: After both counters have begun to count they are both disabled by software at the same time -- the difference in counts multiplied by the on-board clock rate is the time interval.

1.10. Daughter Board

Supports the functions which utilize the daughter boards to enhance the PC-LabCard's capabilities

1.10.1. PCLD-779 Isolated Relay Multiplexer/Amplifier Board

The PCLD-779 is a fully isolated solution, offering precise multi-channel temperature measurements when connected with a PC-LabCard. The driver can support up to 32 A/D channels from PCLD-779's (each PCLD-779 has 8 channels and a CJC channel), but connections must be made as shown below.

The driver offers a simple way for users to implement thermocouple measurement through function calls.

NOTE: Connections between the PCLD-779's channels and the PC-LabCard's channels (assume using PCL-711B).

PCLD-779		PCL-711B	PCL-711B
Board	Channels	CJC CH	DATA CH
1	0-7	1	0
2	8-15	3	2
3	16-23	5	4
4	24-31	7	6

The PC-LabCard system can be expanded with up to 4 PCLD-779 boards by using the PCLD-774 expansion board

CHAPTER 2. PARAMETER TABLE

2.1. Parameter Table

Parameter Table

Name	Size	Index
Board number	1 word	Param[0]
Base I/O address	1 word	Param[1]
DMA channel of buffer A	1 word	Param[2]
DMA channel of buffer B	1 word	Param[3]
Interrupt level	1 word	Param[4]
Pacer rate	2 words	Param[5]
Trigger mode	1 word	Param[7]
Cyclic or Non-cyclic	1 word	Param[8]
Reserve	1 word	Param[9]
A/D Data Buffer A's address	2 words	Param[10]
A/D Data Buffer B's address	2 words	Param[12]
A/D conversion number	1 word	Param[14]
A/D start channel	1 word	Param[15]
A/D stop channel	1 word	Param[16]
Overall gain code	1 word	Param[17]
Gain code array pointer	2 words	Param[18]
D/A Buffer A's address	2 words	Param[20]
D/A Buffer B's address	2 words	Param[22]
D/A conversion number	1 word	Param[24]
D/A start channel	1 word	Param[25]
D/A stop channel	1 word	Param[26]
D/I Buffer A's address	2 words	Param[27]
D/I Buffer B's address	2 words	Param[29]
D/I input number	1 word	Param[31]
D/I port selection	1 word	Param[32]
D/O Buffer A's address	2 words	Param[33]
D/O Buffer B's address	2 words	Param[35]
D/O output number	1 word	Param[37]
D/O port selection	1 word	Param[38]
T/C channel A	1 word	Param[39]
T/C channel B	1 word	Param[40]
T/C time-tick A	2 words	Param[41]
T/C time-tick B	2 words	Param[43]

Name	Size	Index
Error number	1 word	Param[45]
Return value 0	1 word	Param[46]
Return value 1	1 word	Param[47]
Return value 2	1 word	Param[48]
Return value 3	1 word	Param[49]
Daughter Board Type	1 word	Param[50]
Transfer Mode (S/W or INT)	1 word	Param[51]
CJC data buffer's address	2 words	Param[52]
Reserved	1 word	Param[54]
Reserved	1 word	Param[55]

2.2. Parameter Descriptions

Param[0] 0 = Specify Number one Card.
1 = Specify Number two Card.

The software driver supports up to two PC-LabCards at one time. Set Param[0] to tell the driver which card is specified.

Param[1] PC-LabCard I/O address can be anywhere from 200 to 3F0(Hex). The base address can be set to 200 or 210, 230 or 240 3F0

Param[2] DMA channel of buffer A.

The DMA channel number for transfer data to/from buffer A.

Param[3] DMA channel of buffer B.

The DMA channel number for transfer data to/from buffer B.

Param[4] Set the PC-LabCard's IRQ level from 2 to 7.

Param[5] C1 = 16-bit data, from 2 to 65535

Param[6] C2 = 16 bit data, from 2 to 65535

C1 and C2 are both used to program the PC-LabCard's pacer rate using the following formula

(In the case of PCL-711B)
Pacer rate = 2 MHz/(C1*C2)

If using PC-LabCard other than the PCL-711B, replace 2 MHz with the appropriate value for your card. For information, refer to your hardware manual

NOTE: When using the BASIC language, negative numbers must be used to represent integer data over 32767

negative number = integer - 65536.

For example, you need to pass 45000 as a input parameter to BASIC function 45000 - 65536 = -20536 So, you have to use -20536 rather than 45000 as a parameter for the BASIC function

Param[7] 0 = Internal trigger (pacer trigger).
1 = External trigger (not allowed for D/I function)

Param[8] 0 = Non-cyclic conversion mode
1 = Cyclic conversion mode

When you start conversion, data will be stored to the beginning of the data buffer, and continue to be stored sequentially You will have to specify the number of conversions you want to perform as is defined in the table. For example, you might want to perform 100 A/D conversions. In non-cyclic mode, the driver will stop at the 100th conversion. But in cyclic mode, the driver will not stop after the 100th conversion. Data will continue to be stored in sequential order from the beginning of the A/D data buffer, repeating the cycle until the Stop function command has been issued.

NOTE: Cyclic mode is not supported by Software Data Transfer.

Param[9] Reserved

Param[10] Offset address for A/D data buffer A.

Param[11] Segment address for A/D data buffer A.

Param[12] Offset address for A/D data buffer B.

Param[13] Segment address for A/D data buffer B.

NOTE: For C or PASCAL, use their built-in memory allocation functions to allocate sufficient memory for buffers A and B. These memory allocation functions will return the offset and segment addresses. Save them to Param[10] through Param[13]. If buffer B is not used, be sure that you set Param[12] and Param[13] as 0.

Because BASICA and Quick BASIC do not provide memory allocation functions, you will have to assign explicit segment addresses for each buffer. If you assign a segment address as 0, then the driver will use the current data segment (DS) for buffers A and B. If buffer B is not used, be sure that you set Param[12] and Param[13] as 0

Param[14] This parameter sets the A/D conversion number. The range is from 1 to 32767

Param[15] Sets the A/D start channel number.

Param[16] Sets the A/D stop channel number.

Param[17] The driver allows you to set all the A/D channels to the same amplification gain. Param[17] sets the A/D gain code for all channels Remember that individual amplification gains can be set for each channel, defined in the gain array table
This parameter is used for only setting all A/D channels to the same amplification gain
Set Param[17] to FF (Hex) to cause the driver to refer to the gain array table.

Param[18] Offset address for the gain array table

Param[19] Segment address for the gain array table.

Param[20] Offset address for the D/A data buffer A.

Param[21] Segment address for the D/A data buffer A.

Param[22] Offset address for the D/A data buffer B

Param[23] Segment address for the D/A data buffer B.

Param[24] This parameter sets the D/A conversion number The range is from 1 to 32767.

Param[25] Sets the D/A conversion start channel number.

Param[26] Sets the D/A conversion stop channel number.

Param[27] Offset address for the D/I data buffer A.

Param[28] Segment address for the D/I data buffer A.

Param[29] Offset address for the D/I data buffer B

Param[30] Segment address for the D/I data buffer B

Param[31] Sets the number of digital inputs to be performed. All data will be stored in the D/I data buffer The range is from 1 to 65535.

Param[32] Sets the desired digital input port.

0 = Port 0 (DI7 to DI0)
1 = Port 1 (DI15 to DI8)

Param[33] Offset address for the D/O data buffer A.

Param[34] Segment address for the D/O data buffer A.

Param[35] Offset address for the D/O data buffer B

Param[36] Segment address for the D/O data buffer B

Param[37] Sets the number of digital outputs to be performed. All data will be accessed from the D/O data buffer. The range is from 1 to 65535

Param[38] Sets the D/O port

0 = Port 0 (DO7 to DO0)
1 Port 1 (DO15 to DO8)

Param[39] T/C channel A

Param[40] T/C channel B

Param[41] T/C time-tick A

Param[43] T/C time-tick B

Param[45] Error number

Param[46] Return value 0.

Param[47] Return value 1

Param[48] Return value 2

Param[49] Return value 3

Param[50] Daughter Board Type
1 for PCLD-779

Param[51] Transfer mode for daughter board.

0 : Software
1 DMA (not supported by PCLD-779).
2 Interrupt.

Param[52] Offset address for CJC data buffer

Param[53] Segment address for CJC data buffer

Param[54] Reserved.

Param[55] Reserved

CHAPTER 3. DRIVER FUNCTIONS

This chapter describes functions supported by the software driver. The software driver is a **TSR (Terminate and Stay Resident)** program. You must install the driver before you can use the following functions. Each PC-LabCard has its own driver, loaded by typing the appropriate filename at the DOS prompt. For the case of using PC-LabCard PCL-718, just type "PCL-718" at DOS prompt, and press enter to load the program.

Keep in mind that not all the following functions are supported by every software driver. Because some functions require special hardware which is available only on some advanced PC-LabCards.

3.1. Function List

3.1.1. Special Function Calls

Func 0 Get Error Message.
Func 1 Reserve.
Func 2 Get Driver Version Number
Func 3 Driver Initialization

3.1.2. A/D Function Calls

Func 4 A/D Initialization
Func 5 Perform A/D conversion with software data transfer.
Func 6 Perform A/D conversion with DMA data transfer.
Func 7 Get Func 6's operational status.
Func 8 Stop Func 6
Func 9 Perform A/D conversion with interrupt data transfer.
Func 10 Get Func 9's operational status
Func 11 Stop Func 9.

3.1.2.1. Block Channel Scan Function Calls

Func 100 Block channel scan initialization
Func 101 Perform Block channel scan with software data transfer
Func 102 Reserved
Func 103 Reserved
Func 104 Reserve
Func 105 Perform Block channel scan with interrupt data transfer
Func 106 Get Func 105 status.
Func 107 Stop Func 105

3.1.3. D/A Function Calls

Func 12 D/A Initialization
Func 13 Perform D/A conversion with software data transfer
Func 14 Perform D/A with DMA data transfer
Func 15 Get Func 14's operational status.
Func 16 Stop Func 14.
Func 17 Perform D/A conversion with interrupt data transfer.
Func 18 Get Func 17's operational status.
Func 19 Stop Func 17.

3.1.4. Digital Input Function Calls

Func 20 D/I initialization
Func 21 Perform digital input with software data transfer.
Func 22 Perform digital input with DMA data transfer.
Func 23 Get Func 22's operational status.
Func 24 Stop Func 22
Func 25 Perform digital input with interrupt data transfer.
Func 26 Get Func 25's operational status.
Func 27 Stop Func 25

3.1.5. Digital Output Function Calls

Func 28 D/O initialization
Func 29 Perform digital output with software data transfer
Func 30 Read back current digital output status.
Func 31 Perform digital output with DMA data transfer
Func 32 Get Func 31's operational status
Func 33 Stop Func 31
Func 34 Perform digital output with interrupt data transfer
Func 35 Get Func 34's operational status
Func 36 Stop Func 34.

3.1.6. Timer/Counter Function Calls

Func 37 Timer initialization
Func 38 Timer interrupt enable.
Func 39 Timer interrupt disable.
Func 40 Frequency measurement start
Func 41 Get the Func 40 status.
Func 42 Stop Func 40.
Func 43 Event count start.
Func 44 Read event count.
Func 45 Stop event count Func 43.
Func 46 Pulse output start.
Func 47 Pulse output stop.
Func 48 One-shot pulse output.
Func 49 Time interval measurement start.
Func 50 Get Func 49 status.
Func 51 Stop Func 49.

3.1.7. Daughter Board Function Calls

Func 96 Daughter board A/D initialization.
Func 97 Perform daughter board A/D conversion with software or interrupt data transfer.
Func 98 Get Func 97 status
Func 99 Stop Func 97.

3.2. Function description

3.2.1. Special Function Calls

Func 0: Get Error Message.

This function returns a zero-terminated text string pointer corresponding to an error code. The zero-terminated text string is a text string appended with numeric zero at end.

Parameters used.
param[45]: Error code
param[46]: Offset of address of the string pointer.
param[47]: Segment of address of the string pointer.

Return data
param[46]: Offset of address of the string pointer.
param[47]: Segment of address of the string pointer.

Func 2. Get Driver Version Number.

This function returns the current version of the driver as well as the version of this Driver Specification.

Parameters used:
param[45]: Error code
param[46]: Driver specification version number.
param[47]: Driver version number.

Return data
param[45]: Error code
param[46] Driver specification version number.
param[47]. Driver version number.

Func 3: Driver Initialization.

The PC-LabCard is initialized according to the parameter's definitions. It will stop all functions, release all resources. It should be called before any other function

Parameters used:
param[0]: Board number.
param[1]: Base I/O address.
Param[2]: DMA channel of buffer A.
Param[3]: DMA channel of buffer B.
param[4]: Interrupt level.
param[5]: Pacer rate.
param[7]: Trigger mode.
param[8]: Cyclic or non-cyclic mode.

Return data:
param[45]: Error code.

3.2.2. A/D Function Calls

Parameters used:
param[0]: Board number.
param[1]: Base I/O address.
Param[2]: DMA channel of buffer A.
Param[3]: DMA channel of buffer B.
param[4]: Interrupt level.
param[5]: Pacer rate.
param[7]. Trigger mode.
param[8]: Cyclic or non-cyclic mode.
param[10] A/D Data Buffer A's address
param[12] A/D Data Buffer B's address
param[14] A/D conversion number
param[15] A/D start channel
param[16] A/D stop channel
param[17] Overall gain code
param[18] Gain code array address

Return data
param[45] Error code
param[46] Return value #0
param[47] Return value #1

Func 4: A/D Initializing

This function is used to initialize the PC LabCard's A/D functions according to the above parameter's setting. It should be called before any other A/D function.

Return data
param[45]: Error code

Func 5: Perform A/D conversion with software data transfer.

This function will perform A/D conversion N times with software data transfer. It will not return until the Nth conversion has been completed. The value of 'N' is specified at param[14]. For this function, only buffer A can be used as a data buffer and Cyclic mode is not allowed.

It should be noted that the time interval between each A/D conversion is dependent on the clock rate (internal or external). The driver triggers an A/D conversion and waits for the next clock pulse to do the next conversion.

Return data.
param[45]: Error code.

Func 6: Perform A/D conversion with DMA data transfer.

This function will perform A/D conversion N times with DMA data transfer. The DMA data transfer takes place in the background which will not be stopped until the Nth conversion is completed or your program executes Func 8 to stop the process. Under cyclic mode, use Func 8 to stop the process. The value 'N' is specified in param[14].

After executing this function, it is necessary to check the status of the operation by calling Func 7.

Return data
param[45] Error code

Func 7: Get Func 6's operation status.

Since Func 6 works in the background, you can issue this function to check its operational status.

If you are using double-buffered mode, check bit 1 of param[46] to see which buffer is in progress. Because it takes time to switch between buffers (depending on your CPU's performance), some of your data may become lost during the switching period. Check bit 2 of param[46] to see if the data buffer changed since the last call of this function. Bit 2 of param[46] is also cleared after calling this function.

Return data.
param[45]: Error code.
param[46]:

Bit 0: 0 = is completed.
1 = is not completed.
Bit 1: 0 = is using A/D buffer A.
1 = is using A/D buffer B.
Bit 2: 0 = data buffer not changed yet.
1 = data buffer changed.

param[47]: Returns the current conversion count number, which starts from 0.

Func 8: Stop Func 6.

Use this function to stop Func 6, N time, pacer triggered A/D conversion with DMA data transfer.

Refer to Func 7 for the meaning of returned data.

Return data.
param[45] Error code.
param[46]:

Bit 0: 0 = is completed.
1 = is not completed.
Bit 1: 0 = is using A/D buffer A.
1 = is using A/D buffer B.
Bit 2: 0 = data buffer not changed yet.
1 = data buffer changed.

param[47]: Returns the current conversion count number.

Func 9: Perform A/D conversion with interrupt data transfer.

This function will perform A/D conversion N times with interrupt data transfer. The interrupt data transfer takes place in the background and will not be stopped until the Nth conversion is completed or your program executes Func 11 to stop the process. Under cyclic mode, use Func 11 to stop it. The value 'N' is specified in param[24].

It is necessary after executing this function to check the operational status by calling Func 10.

Return data:
param[45]: Error code.

Func 10: Get Func 9's operation status.

Since Func 9 works in the background, you can issue this function to check its operational status.

If you are using double-buffered mode, check bit 1 of param[46] to see which buffer is in progress. Because it takes time to switch between buffers (depending on your CPU's performance), some of your data may become lost during the switching period. Check bit 2 of param[46] to see if the data buffer has changed since the last call of this function. Bit 2 of param[46] is also cleared after calling this function.

Return data:
param[45]: Error code.

param[46]:

Bit 0: 0 = is completed.
1 = is not completed.
Bit 1: 0 = is using A/D buffer A.
1 = is using A/D buffer B.
Bit 2: 0 = data buffer not changed yet.
1 = data buffer changed.

param[47]: Returns the current conversion count number, which starts from 0.

Func 11: Stop Func 9.

Use this function to stop Func 9, 'N' times pacer triggered A/D conversion with interrupt data transfer. Refer to Func 10 for meaning of returned data.

Return data:
param[45]: Error code
param[46]:

Bit 0: 0 = is completed.
1 = is not completed.
Bit 1: 0 = is using A/D buffer A.
1 = is using A/D buffer B
Bit 2: 0 = data buffer not changed yet.
1 = data buffer changed.

param[47]: Returns the current conversion count number.

3.2.2.1. Block Channel Scan Function Calls

Parameters used
param[0]: Board number
param[1]: Base I/O address
Param[2]: DMA channel of buffer A.
Param[3]: DMA channel of buffer B
param[4] Interrupt level
param[5] Pacer rate
param[7]: Trigger mode
param[8]: Cyclic or non-cyclic mode.
param[10] A/D Data Buffer A's address
param[12] A/D Data Buffer B's address
param[14] A/D Conversion number
param[15] A/D Block scan start channel
param[16] A/D Block scan stop channel
param[17] Overall gain code
param[18] Gain code array address
param[45] Error code
param[46] Return value 0
param[47] Return value 1

Func 100: Block channel scan initialization.

This function is used to initialize the PC-LabCard's A/D according to the above parameter's setting for subsequent block channel scan functions.

Return data.
param[45]: Error code.

Func 101: Perform Block channel scan with software data transfer.

Block channel scan makes software triggered scans on every channel of a block. This function will perform block channel scans N times. It will not return until the Nth conversion scan has been completed. The value 'N' is specified in param[14]. Note that the time interval between each Block channel scan is dependent on clock rate (internal or external). The driver makes a scan and waits for next clock pulse to do next scan.

Return data
param[45] Error code

Func 105. Perform Block channel scan with interrupt data transfer.

Block channel scan makes software triggered scans on every channel of a block. This function will perform block channel scan N times. Each block channel scan is activated by an interrupt routine which is triggered by internal or external clock pulse.

The scan takes place in the background and will not stop until the Nth conversion scanning is completed or Func 107 is called to stop the process. The value 'N' is specified in param[14].

In double buffered mode, when switching from one buffer to another, at the time the value 'N' within a block is reached, the next buffer will continue at the next channel until the stop channel value is reached. Then the next cycle begins with the start value.

It is necessary after executing this function to check the operation's status by calling Func 106.

Func 106: Get Func 105 status.

Since Func 105 works in the background, you can issue this function to check its operational status. If you are using double-buffered mode, check bit 1 of param[46] to see which buffer is current. Because it takes time to switch between buffers (depending on your CPU's performance), some of your data may become lost during the switching period. Check bit 2 of param[46] to see if the data buffer changed since the last call of this function. Bit 2 of param[46] is also cleared after calling this function.

Return data:
param[45]: Error code.

param[46]:

Bit 0: 0 = is completed.
1 = is not completed.
Bit 1: 0 = is using D/A buffer A.
1 = is using D/A buffer B.
Bit 2: 0 = data buffer not changed yet.
1 = data buffer changed.

param[47]: Returns the current conversion count number, which starts from 0.

Func 107: Stop Func 105.

Use this function to stop Func 105.
Refer to Func 106 for meaning of returned data.

Return data:
param[45]: Error code.

param[46]:

Bit 0: 0 = is completed.
1 = is not completed
Bit 1: 0 = is using D/A buffer A.
1 = is using D/A buffer B.
Bit 2: 0 = data buffer not changed yet.
1 = data buffer changed.

param[47] Returns the current conversion count number.

3.2.3. D/A Function Calls

Parameters used:
param[0] Board number
param[1]. Base I/O address
Param[2] DMA channel of buffer A.
Param[3]: DMA channel of buffer B
param[4] Interrupt level
param[5] Pacer rate

param[7]: Trigger mode.
param[8]: Cyclic or non-cyclic mode:
param[20] D/A Data Buffer A's address.
param[22] D/A Data Buffer B's address.
param[24] D/A conversion number.
param[25] D/A start channel.
param[26] D/A stop channel.
param[45] Error code.
param[46] Return value #0.
param[47] Return value #1.

Func 12: D/A Initialization

This function is used to initialize the PC-LabCard's D/A according to the above parameter settings. It should be called before any other D/A function.

Return data:
param[45]: Error code.

Func 13: Perform D/A conversion with software data transfer.

This function will perform D/A conversion N times with software data transfer. It will not return until the Nth conversion has been completed. The value of 'N' is specified at param[24]. For this function, only buffer A can be used as a data buffer and Cyclic mode is not allowed.

Note that the time interval between each D/A conversion is dependent on clock rate(internal or external). The driver triggers an D/A conversion and waits for the next clock pulse to do the next conversion.

Return data:
param[45]: Error code.

Func 14: Perform D/A conversion with DMA data transfer.

This function will perform D/A conversion N times with DMA data transfer. The DMA data transfer takes place in the background and will not be stopped until the Nth conversion is completed or your program executes Func 16 to stop the process. Under cyclic mode, use Func 16 to stop the process.

The value 'N' is specified in param[24].

It is necessary after executing this function to check the operational status by calling Func 15.

Return data
param[45]: Error code

Func 15: Get Func 14's operation status.

Since Func 14 works in the background, you can issue this function to check its operational status. If you are using double-buffered mode, check bit 1 of param[46] to see which buffer is in progress. Because it takes time to switch between buffers (depending on your CPU's performance), some of your data may become lost during the switching period. Check bit 2 of param[46] to see if the data buffer changed since the last call of this function. Bit 2 of param[46] is also cleared after calling this function.

Return data
param[45]: Error code.

param[46]

Bit 0: 0 = is completed
1 = is not completed
Bit 1: 0 = is using D/A buffer A.
1 = is using D/A buffer B.
Bit 2: 0 = data buffer not changed yet.
1 = data buffer changed.

param[47] Returns the current conversion count number, which starts from 0

Func 16: Stop Func 14.

Use this function to stop Func 14, 'N' time, pacer triggered D/A conversion with DMA data transfer
Refer to Func 15 for meaning of return data.

Return data:
param[45]: Error code

param[46]

Bit 0: 0 = is completed.
1 = is not completed
Bit 1: 0 = is using D/A buffer A.
1 = is using D/A buffer B.
Bit 2: 0 = data buffer not changed yet
1 = data buffer changed

param[47]: Returns the current conversion count number.

Func 17: Perform D/A with interrupt data transfer.

This function will perform D/A N times conversion with interrupt data transfer. The interrupt data transfer takes place in the background and will not be stopped until the Nth conversion transfer is completed or your program executes Func 16 to stop the process. Under cyclic mode, use Func 16 to stop the process. The value 'N' is specified in param[24].

It is necessary after executing this function to check the operation's status by calling Func 18.

Return data:
param[45]: Error code.

Func 18: Get Func 17's operation status.

Since Func 17 works in the background, you can issue this function to check its operation status.

If you are using double-buffered mode, check bit 1 of param[46] to see which buffer is in progress. Because it takes time to switch between buffers (depending on your CPU's performance), some of your data may become lost during the switching period. Check bit 2 of param[46] to see if the data buffer has changed since the last call of this function. Bit 2 of param[46] is also cleared after calling this function.

Return data:
param[45]: Error code.

param[46]:

Bit 0: 0 = is completed.
1 = is not completed.
Bit 1: 0 = is using D/A buffer A.
1 = is using D/A buffer B.
Bit 2: 0 = data buffer not changed yet.
1 = data buffer changed.

param[47]: Returns the current conversion count number, which starts from 0.

Func 19: Stop Func 17.

Use this function to stop Func 17, 'N' time, pacer triggered D/A conversion with interrupt data transfer. Refer to Func 18 for meaning of return data.

Return data:
param[45]: Error code.

param[46]:

Bit 0: 0 = is completed.
1 = is not completed.
Bit 1: 0 = is using D/A buffer A.
1 = is using D/A buffer B.
Bit 2: 0 = data buffer not changed yet.
1 = data buffer changed.

param[47]: Returns the current conversion count number.

3.2.4. Digital Input Function Calls

Parameters used:
param[0]. Board number.
param[1]: Base I/O address.
Param[2]: DMA channel of buffer A.
Param[3]: DMA channel of buffer B.
param[4]: Interrupt level.
param[5]: Pacer rate.
param[7]: Trigger mode (only internal clock trigger is allowed).
param[8]: Cyclic or non-cyclic mode.

param[27] D/I data buffer A's address
param[29] D/I data buffer B's address
param[31] Number of digital input
param[32] D/I port selection

param[45] Error code
param[46] Return value #0
param[47] Return value #1

Func 20: D/I initialization.

This function is used to initialize the PC-LabCard's D/I operation according to the above parameter's settings. It should be called before any other D/I function. For D/I functions, trigger mode must be set to internal clock (pacer) mode. An external clock signal cannot be used as a trigger source because DI0 is the pin to be connected to the external clock signal.

This conflicts with its digital input function

Return data
param[45] Error code

Func 21: Perform digital input with software data transfer.

This function will perform digital input N times, with software data transfer. It will not return until the Nth conversion input has been completed. The value of 'N' is specified at param[31]. For this function, only buffer A can be used as a data buffer and Cyclic mode is not allowed.

Noted that the time interval between each digital input is dependent on the internal clock rate (pacer). The driver triggers a D/I conversion and waits for the next clock pulse to do the next conversion.

Return data
param[45]. Error code

Func 22: Perform digital input with DMA data transfer.

This function will perform Digital input N times with DMA data transfer. The DMA data transfer takes place in the background which and will not be stopped until the Nth conversion input is completed or your program executes Func 24 to stop the process. Under cyclic mode, use Func 24 to stop it.

The value 'N' is specified in param[31].

It is necessary after executing this function to check the operation's status by calling Func 23.

Return data:
param[45]: Error code

Func 23. Get Func 22's operation status.

Since Func 22 works in the background, you can issue this function to check its operation status. If you are using double-buffered mode, check bit 1 of param[46] to see which buffer is in progress. Because it takes time to switch between buffers (depending on your CPU's performance), some of your data may become lost during the switching period. Check bit 2 of param[46] to see if the data buffer has changed since the last call of this function. Bit 2 of param[46] is also cleared after calling this function.

Return data
param[45]: Error code

param[46]:

Bit 0: 0 = is completed.
1 = is not completed.
Bit 1: 0 = is using D/I buffer A.
1 = is using D/I buffer B.
Bit 2: 0 = data buffer not changed yet.
1 = data buffer changed.

param[47]: Returns the current input count number, which starts from 0.

Func 24: Stop Func 22.

Use this function to stop Func 22, N time, pacer triggered digital input with DMA data transfer.
Refer to Func 23 for meaning of return data.

Return data:
param[45]: Error code.

param[46]:

Bit 0: 0 = is completed.
1 = is not completed.

Bit 1: 0 = is using D/I buffer A.
1 = is using D/I buffer B.
Bit 2: 0 = data buffer not changed yet.
1 = data buffer changed.

param[47]: Returns the current input count number.

Func 25: Perform digital input with interrupt data transfer.

This function will perform Digital N times input with interrupt data transfer. The interrupt data transfer takes place in the background and will not be stopped until the Nth conversion input is completed or your program executes Func 27 to stop the process. Under cyclic mode, use Func 27 to stop data transfer. The value 'N' is specified in param[31].

After executing this function, it is necessary to check the operation's status by calling Func 26.

Return data:
param[45]: Error code.

Func 26: Get Func 25's operational status.

Since Func 25 works in the background, you can issue this function to check its operational status. If you are using double-buffered mode, check bit 1 of param[46] to see which buffer is in progress. Because it takes time to switch between buffers (depending on your CPU's performance), some of your data may become lost during the switching period. Check bit 2 of param[46] to see if the data buffer has changed since the last call of this function. Bit 2 of param[46] is also cleared after calling this function.

Return data:
param[45]. Error code

param[46]:

Bit 0: 0 = is completed.
1 = is not completed.
Bit 1: 0 = is using D/I buffer A.
1 = is using D/I buffer B.
Bit 2: 0 = data buffer not changed yet.
1 = data buffer changed.

param[47]: Returns the current input count number, which starts from 0.

Func 27: Stop Func 25.

Use this function to stop Func 25, N time, pacer triggered digital input with interrupt data transfer. Refer to Func 26 for meaning of return data.

Return data:
param[45]: Error code.

param[46]:

Bit 0: 0 = is completed.
1 = is not completed.
Bit 1: 0 = is using D/I buffer A.
1 = is using D/I buffer B.
Bit 2: 0 = data buffer not changed yet.
1 = data buffer changed.

param[45]: Returns the current input count number.

3.2.5. Digital Output Function Calls

Parameters used:
param[0]: Board number.
param[1]: Base I/O address.
Param[2]: DMA channel of buffer A.
Param[3]: DMA channel of buffer B.
param[4]: Interrupt level
param[5]. Pacer rate
param[7]: Trigger mode

param[8] Cyclic or non-cyclic mode

param[33] D/O data buffer A's address
param[35] D/O data buffer B's address
param[37] Number of digital output
param[38] D/O port selection

param[45] Error code
param[46] Return value #0
param[47] Return value #1

Func 28: D/O initialization.

This function is used to initialize the PC-LabCard's D/O according to the above parameter's setting. It should be called before any other D/O function.

Return data:
param[45]: Error code.

Func 29: Perform digital output with software data transfer.

This function will perform digital output N times with software data transfer. It will not return until the Nth conversion output has been completed. The value of 'N' is specified at param[37]. For this function, only buffer A can be used as a data buffer and Cyclic mode is not allowed.

Note that the time interval between each digital output is dependent on clock rate (internal or external). The driver triggers a D/O conversion and waits for the next clock pulse to do the next conversion.

Return data:
param[45]: Error code.

Func 30: Read back current digital output status.

This function reads back the digital status of output port. Usually this routine will retrieve this value from driver resident table that was saved when the last digital function was executed. However, those digital devices that allow direct reading of the current output port content should do so.

Return data:
param[45]: Error code.

param[46]: output port data.

Func 31: Perform digital output with DMA data transfer.

This function will perform Digital output N times with DMA data transfer. The DMA data transfer takes place in the background and will not be stopped until the Nth input is completed or your program executes Func 33 to stop the process. Under cyclic mode, use Func 33 to stop it. The value 'N' is specified in param[37].

After executing this function, it is necessary to check the operation's status by calling Func 32.

Return data:
param[45]: Error code.

Func 32: Get Func 31's operation status.

Since Func 31 works in the background, you can issue this function to check its operational status.

If you are using double-buffered mode, check bit 1 of param[46] to see which buffer is in progress. Because it takes time to switch between buffers (depending on your CPU's performance), some of your data may become lost during the switching period. Check bit 2 of param[46] to see if the data buffer has changed since the last call of this function. Bit 2 of param[46] is also cleared after calling this function.

Return data:
param[45]: Error code.

param[46]:

Bit 0: 0 = is completed.
1 = is not completed.
Bit 1: 0 = is using D/O buffer A.
1 = is using D/O buffer B.
Bit 2: 0 = data buffer not changed yet.
1 = data buffer changed.

param[47]: Returns the current output count number, which starts from 0.

Func 33: Stop Func 31.

Use this function to stop Func 31, N time, pacer triggered digital output with DMA data transfer. Refer to Func 32 for the meaning of return data.

Return data:
param[45]: Error code.

param[46]:

Bit 0: 0 = is completed.
1 = is not completed.
Bit 1: 0 = is using D/I buffer A.
1 = is using D/I buffer B.
Bit 2: 0 = data buffer not changed yet.
1 = data buffer changed.

param[47] Returns the current output count number.

Func 34: Performs digital output with interrupt data transfer.

This function will perform Digital output N times with interrupt data transfer. The interrupt data transfer takes place in the background and will

not be stopped until the Nth conversion output is completed or your program executes Func 36 to stop the process. Under cyclic mode, use Func 36 to stop it. The value 'N' is specified in param[37].

After executing this function, it is necessary to check the operation's status by calling Func 35.

Return data:

param[45]: Error code.

Func 35: Get Func 34's operation status.

Since Func 34 works in the background, you can issue this function to check its operation status.

If you are using double-buffered mode, check bit 1 of param[46] to see which buffer is in progress. Because it takes time to switch between buffers (depending on your CPU's performance), some of your data may become lost during the switching period. Check bit 2 of param[46] to see if the data buffer has changed since the last call of this function. Bit 2 of param[46] is also cleared after calling this function.

Return data:
param[45]: Error code.

param[46]:

Bit 0: 0 = is completed.
1 = is not completed.
Bit 1: 0 = is using D/O buffer A.
1 = is using D/O buffer B.
Bit 2: 0 = data buffer not changed yet.
1 = data buffer changed.

param[47]: Returns the current output count number, which starts from 0.

Func 36: Stop Func 34.

Use this function to stop Func 34, N time, pacer triggered digital output with interrupt data transfer. Refer to Func 35 for meaning of return data.

Return data
param[45]: Error code.

param[46]

Bit 0: 0 = is completed.
1 = is not completed
Bit 1: 0 = is using D/O buffer A
1 = is using D/O buffer B.
Bit 2: 0 = data buffer not changed yet.
1 = data buffer changed

param[47]: Returns the current output count number.

3.2.6. Timer/Counter Function Calls

Parameters used:
param[0]: Board number.
param[1]: Base I/O address.

param[39]: T/C channel A
param[40]: T/C channel B
param[41]: T/C time-tick A
param[43]: T/C time-tick B

param[45]: Error code
param[46]: Return value #0
param[47]: Return value #1

Func 37: Timer initialization.

This function is used to initialize the PC-LabCard's timer to a well defined state according to the above parameter's setting for subsequent

timer operations

Return data.
param[45]: Error code

3.2.6.1. Timer Interrupt

Parameters used:
param[0]: Board number.
param[1]: Base I/O address

param[39]: Timer interrupt channel used for this function
param[41]: Timer ticks (2 words).
param[45]: Error code

Func 38: Timer interrupt enable.

This function enables the PC-LabCard to generate hardware timer interrupts at the specified rate in units of "timer tick resolution" units. The rate is a double precision integer value in the parameter table, param[41]. The driver will install an interrupt handling routine for hardware timer interrupts that will execute a long call to TI vect : (interrupt vector 64h and use "RETF" to return). The TI vector is supplied by an application program. This way an application can do something periodically.

For example, if the timer tick duration is 1.25 microseconds, then a 40 Hz timer rate could be specified by a parameter of 20,000 ticks. The timer tick duration is determined by the clock input to the Timer/Counter chip. Note that overrun error cannot be detected and you need to do some connections between different pins for this specific function. Please refer to User's Manual for detailed hardware figure.

Timer interrupt frequency = 1 / ((timer tick duration) * (count))
count ticks = high word * low word. If high word is 0, use low word value only.

Return data:
param[45]: Error code.

Func 39: Timer interrupt disable.

This function disables Func 38, the timer interrupt and releases used resource.

Return data:
param[45]: Error code.

3.2.6.2. Frequency Measurement

Parameters used:
param[0]: Board number.
param[1]: Base I/O address.

param[39]: Frequency input timer channel.
param[41]: Count for gate counter (2 words).
param[45]: Error code.

NOTE: The "gate counter" is implicitly fixed at a timer channel and the distinct channel number depends on the card you use. Refer to the User's Manual for the specific channel number.

Func 40: Frequency measure start

This function sets up timer channels to start measuring frequency. Keep in mind that the frequency value is not available until the gating period expires. The frequency value is returned by calling Func 41.

Return data:
param[45]: Error code

Func 41: Get the Func 40 status.

Check param[48] to see if the gating interval used for frequency measurement has expired and return a 32-bit frequency count value. Keep in mind that the frequency value is invalid until "done" status is returned.

This count value must be used by the application program to calculate the actual frequency.

frequency = (measured count)/((timer tick duration)*(gate count))

Return data:
param[45]: Error code.
param[46]: Frequency count low word.
param[47]: Frequency count high word.
param[48]: Status flag (0 = not done; not 0 = done).

Func 42: Stop Func 40.

This function is to abort a current frequency measurement process issued by Func 40 without waiting for the end of the gating period. Keep in mind that this does not have to be done if the gate period has expired normally.

Return data: None.

3.2.6.3. Event Counting

Parameters used:
param[0]: Board number.
param[1]: Base I/O address.

param[39]: Event input timer channel.
For the PCL-718,PCL-818,PCL-812, or PCL-812PG, only channel 0 can be selected.
param[45]: Error code.

Func 43: Event count start.

This function sets up a timer channel to start counting an externally generated "event" (for example, pulses). Keep in mind that the current count value is available for reading anytime, but may change until the Event Count Stop function is subsequently executed.

Return data
param[45] Error code

Func 44, Read event count.

This function returns current count from the specified event count channel. Keep in mind that this value may be read at any time after the Event Count Start function, Func 43, has been executed.

Return data:
param[45] Error code.
param[46] Event count low word.
param[47]. Event count high word

Func 45, Stop event count Func 43.

This function stops the current event counting process on the specified counter channel and also returns the final event count. An error may be generated if the counter has overflowed

Return data:
param[45]: Error code.
param[46]: Event count low word.
param[47]: Event count high word.

3.2.6.4. Pulse Output

Parameters used:
param[0]: Board number.
param[1]: Base I/O address.

param[39]: Pulse output timer channel.
param[41]: "Period" of pulse output in timer ticks (2 words).
Period ticks = high word * low word.
param[43]: "On count" in timer ticks (2 words).
Period" high word must equal "On count" high word.
param[45]: Error code.

Func 46, Pulse output start.

This function sets up timer channel to start generating a continuous stream of pulses with the specified period and duty cycle. Pulses will continue until the PULSE OUTPUT STOP function (Func 47) is called. The "period" parameter is used, along with the driver supplied "timer tick duration" (in microseconds), to specify the pulse rate applied to the input of the specified timer channel. The "on count" parameter specifies the number of ticks during which the output pulse should be high, while the "off count", determined by "period" -"on count", specifies the number of ticks when the pulse is low. The cycle time of the output pulse is:

(period count) * (time tick duration)

The duty cycle is determined by the ratio of the "on count" to the "period"

Return data:
param[45]: Error code.

Func 47: Pulse output stop.

This function turns off pulses that are being generated from a previously executed PULSE OUTPUT START function.

Return data. None.

Func 48: One shot pulse output.

Parameters used:
param[0]: Board number.
param[1]: Base I/O address.

param[39]: Pulse output timer channel.
param[43]: "On count" in timer ticks(2 words).
On count ticks = high word * low word. If high word is 0, use low word value only.
param[45]: Error code.

This function sets up a timer channel to generate a single pulse with the specified "on count" parameter.

Return data:
param[45]: Error code.

3.2.6.5. Time Interval Measurement

Parameters used:
param[0]: Board number.
param[1]: Base I/O address.

param[39]: Counter Channel #1
param[40]: Counter Channel #2
If channel 1 = channel 2, only measure first two pulses.
param[45]: Error code.

Func 49: Time interval measurement start.

This function sets up the counters to start measuring the time interval between two externally generated pulses.

Return data.
param[45]: Error code.

Func 50: Get Func 49 status.

This function returns a status word and timer interval count value. The status word indicates which pulse or pulses have been detected as well as the order of detection. The count is valid only when the status is either 3 or 4.

The application program may convert the count into real units (in microseconds) by using the formula:

time = (count) * (time tick duration)

Return data:
param[45]: Error code.
param[46]: Time interval count low word.
param[47]: Time interval count high word.

param[48]: Status
0: neither pulse detected.
1: first pulse detected, but not second.
2: second pulse detected, but not first.
3: both pulses detected, first channel detected first.
4: both pulses detected, second channel detected first.

Func 51: Stop Func 49.

This function aborts the current time interval measurement process without waiting for both external pulses to be recognized. Note that this does not need to be done if the time interval measurement proceeds normally.

Return data:
param[45]: Error code.

3.2.7. Daughter Board Function Calls

Parameters used:
param[0]: Board number.
param[1]: Base I/O address.
Param[2]: DMA channel of buffer A.
Param[3]: DMA channel of buffer B.
param[4]: Interrupt level.
param[5]: Pacer rate.
param[7]: Trigger mode.
param[8]: Cyclic or non-cyclic mode.

param[10] A/D Data Buffer A's address
param[12] A/D Data Buffer B's address
param[14] A/D conversion number
param[15] Daughter board start channel
param[16] Daughter board stop channel

param[17] Overall gain code
param[18] Gain code array address

param[45] Error code
param[46] Return value #0
param[47] Return value #1

param[50] Daughter board type
1 - PCLD-779

param[51] Transfer mode
0 Software
1 DMA (not supported by PCLD-779)
2 . Interrupt

param[52] Daughter board CJC data buffer address (Offset).
param[53] Daughter board CJC data buffer address (Segment).

There is only one CJC data channel The CJC data buffer should be stored relative to the channel number's data

Func 96: Daughter board A/D initialization.

This function is used to initialize the PC-LabCard's A/D and daughter board according to the above parameter's setting.

Returned data.
param[45]: Error code.

Func 97: Performs daughter board A/D conversion with software or interrupt data transfer.

This function will perform A/D conversion N times with software (param[51]=0) or interrupt (param[51]=2) data transfer. The interrupt data transfer takes place in the background and will not be stopped until the Nth conversion is completed or your program executes Func 99 to stop the process. Under cyclic mode, use Func 99 to stop it. The value 'N' is specified in param[14].

After executing this function, it is necessary to check the operation's status by calling Func 98

Return data
param[45]: Error code

Func 98: Get Func 97 status.

If Func 97 is working under interrupt data transfer mode, you can issue this function to check its operational status. If you are using double-buffered mode, check bit 1 of param[46] to see which buffer is in progress. Because it takes time to switch between buffers (depending on your CPU's performance), some of your data may become lost during the switching period Check bit 2 of param[46] to see if the data buffer has changed since the last call of this function. Bit 2 of param[46] is also cleared after calling this function.

Return data:
param[45]: Error code.

param[46]:

Bit 0: 0 = is completed.
1 = is not completed.
Bit 1: 0 = is using A/D buffer A.
1 = is using A/D buffer B.
Bit 2: 0 = data buffer not changed yet.
1 = data buffer changed.

param[47]: Returns the current conversion count number, which starts from 0.

Func 99: Stop Func 97.

If Func 97 is working under interrupt data transfer mode, use this function to stop Func 97. Refer to Func 98 for the meaning of data returned.

Return data:
param[45]: Error code.

param[46]:

Bit 0: 0 = is completed.
1 = is not completed.
Bit 1: 0 = is using A/D buffer A.
1 = is using A/D buffer B.
Bit 2: 0 = data buffer not changed yet.
1 = data buffer changed.

param[47]: Returns the current conversion count number.

CHAPTER 4. LANGUAGE INTERFACES

This chapter provides information regarding how to program your application program to work with the software driver. Language interfaces are provided for application programs to communicate with the software driver. Language interfaces supported include **BASICA, Quick BASIC 4.0, and 4.5, Microsoft C, Turbo C, Borland C++, Microsoft PASCAL, and Turbo PASCAL.** Before running the application program, you must load the driver into the system's memory.

4.1. BASICA

The following program example provides you with the appropriate procedures to load the language interface for **BASICA** and **GWBASIC version 2.02.**

```
' EXAMPLE (using PC-LabCard PCL-718)!

100 CLEAR 49152!
110 DEF SEG = 0
120 SEG = 256 * PEEK(&H511) + PEEK(&H510)
130 SG = SG + 491521 / 16
140 DEF SEG = SG
150 BLOAD "718BAS.BIN", 0
```

For PC-LabCards other than the PCL-718, you should replace "718BAS.BIN" with the filename for the card you are using.

4.2. GWBASIC (version 3.20)

The following program example provides you with the appropriate procedures to load the language interface for GWBASIC version 3.20 and later.

*** EXAMPLE (using PC-LabCard PCL-718):**

```
110 'LOAD 718BAS.BIN DRIVER TO AN OUTSIDE AREA
120 DEF SEG= &H5000'DEFINE OUTSIDE AREA
130 BLOAD "718BAS.BIN"
140 'END OF DRIVER LOADING
```

For PC-LabCards other than the PCL-718, you should replace "718BAS.BIN" with the filename for the card you are using.

4.3. QuickBASIC 4.0 and 4.5

The following program example provides you with the appropriate procedures to load the language interface for QuickBASIC 4.0 or 4.5.

*** EXAMPLE 1 (using PC-LabCard PCL-718):**

```
QB filename /L 718QB.QLB
```

*** EXAMPLE 2 (using PC-LabCard PCL-718):**

```
QB /L 718QB.QLB
```

For PC-LabCards other than the PCL-718, you should replace "718BAS.QLB" with the filename for the card you are using.

4.4. Microsoft C

The following examples show you how to compile and link the interface for different mode using the Microsoft C.
Assuming you are using LabCard PCL-718. For other PC-LabCards, you should replace "718" with the appropriate number for the card you are using.

Small Mode:	Compile:cl /AS /c file.c Link:link file+718CS.LIB;
Compact Mode:	Compile:cl /AC /c file.c Link:link file+718CC.LIB;
Medium Mode:	Compile:cl /AM /c file.c Link:link file+718CM.LIB;
Large Mode:	Compile:cl /AL /c file.c Link:link file+718CL.LIB;

4.5. Turbo C

The following examples show you how to compile and link the interface for different modes using Turbo C.
This assumes you are using a LabCard PCL-718. For other PC-LabCards, you should replace "718" with the appropriate number for the card you are using.

4.5.1. DOS Command Line

Small Mode:	TCC -ms file.c 718cs.lib
Compact Mode:	TCC -mc file.c 718cc.lib
Medium Mode:	TCC -mm file.c 718cm.lib
Large Mode:	TCC -ml file.c 718cl.lib

4.5.2. Integrated Development Environment

You will need to use a general text editor to create a project file with the extension name "PRJ", for example 718.PRJ, which contains the corresponding mode interface and your program list.

Small Mode:	the project file should contain 718cs.lib
Compact Mode:	the project file should contain 718cc.lib
Medium Mode:	the project file should contain 718cm.lib
Large Mode:	the project file should contain 718cl.lib

4.6. Borland C++

The following examples show you how to compile and link the interface for different modes using the Borland C++.
This assumes you are using LabCard PCL-718. For other PC-LabCards, you should replace "718" with the appropriate number for the card you are using.

4.6.1. DOS Command Line

Small Mode:	BCC -ms -c file.c 718cs.lib
Compact Mode:	BCC -mc -c file.c 718cc.lib
Medium Mode:	BCC -mm -c file.c 718cm.lib
Large Mode:	BCC -ml -c file.c 718cl.lib

4.6.2. Integrated Development Environment

In Borland C++'s integrated environment, just pick the "Project" menu to create a project file and add the corresponding mode interface into it. For the case of the PCL-718, 718cs.lib corresponds to Small Mode; 718cc.lib with Compact Mode; 718cm.lib with Medium Mode; and 718cl.lib corresponds to Large Mode.

4.7. Microsoft PASCAL

The following examples show you how to compile and link the language interface using Microsoft PASCAL.
This assumes you are using LabCard PCL-718. For other PC-LabCards, you should replace "718" with the appropriate number for the card you are using.

*** EXAMPLE:**

```
Compile:
pas1 demoxx
pas2
Link:
link demoxx,,,718msp+pascal;
```

4.8. Turbo PASCAL

The following examples show you how to compile and link the language interface using Turbo PASCAL by adding certain statements in your program. This assumes you are using LabCard PCL-718. For other PC-LabCards, you should replace "718" with the appropriate number for the one you are using.

*** EXAMPLE 1:**

```
program main;
uses Crt;
{$F+}
{$L 718tpf} { use as far call }
```

*** EXAMPLE 2:**

```
program main;
uses Crt;
{$F-}
{$L 718tpn} { use as near call }
```

APPENDIX A. ERROR MESSAGES

No	Error Message
0	No error.
1	No Device.
2	Driver Not Installed.
3	Board Not Initialized Error.
4	Function Not Supported.
5	Function Out of Bounds.
6	Invalid Board Number.
7	Invalid Error Number.
8	Set 8254 Counter Data < 2 .
9	No Data Buffer For Transfer.
10	Expand Board Not Support DMA Transfer.
11	Odd Buffer Pointer Error.
12	Invalid Channel Number(s) Error.
13	Use Gain Array But No Gain Array Pointer Error.
14	Interrupt level In Use Error.
15	Interrupt Overlap Error.
16	This IRQ level Not Supported.
17	DMA Active Error.
18	A/D Not Initialized Error.
19	Non Valid A/D Channel Number(s) Error.
20	D/A Not Initlaized Error.
21	No D/A Module In proper Slot.
22	Digital Input Not Initialized Error.
23	Digital Output Not Initialized Error.
24	No DIO Module In proper Slot.
25	Internal Clock Conflict Error.
26	Timer Not Initialized Error.
27	Frequency Measurement Not Active Error.
28	Frequency Measurement Timer Overrun Error.
29	Event Counting Not Active Error.
30	Event Counting Overrun Error.
31	Pulse Output Not Active Error.
32	Pulse Output Active Error.
33	Async Time Measurement Not Active Error.
34	Async Time Measurement Timer Overrun Error.
35	Async Time Measurement Active Error.
36	Invalid multi-channel.
37	DMA channel error.
38	Transfer Type 0:Software, 2:Interrupt.
39	Card type error.
40	Invalid number of conversions (valid No. 1-32767).
41	Trigger mode error (digital input can use pacer trigger only).
42	Base address error

APPENDIX B. FUNCTIONS SUPPORTED BY EACH PC-LabCard DRIVER.

B.1. PCL-711B

The '*' mark identifies functions supported by the PCL-711B driver.

```
*Func 0   Get Error Message.
*Func 2   Get Driver Version Number.
*Func 3   Driver Initialization.

*Func 4   A/D Initialization
*Func 5   Perform A/D conversion with software data transfer.
 Func 6   Perform A/D conversion with DMA data transfer.
 Func 7   Get Func 6's operation status.
 Func 8   Stop Func 6.
*Func 9   Perform A/D conversion with interrupt data transfer.
*Func 10  Get Func 9's operation status.
*Func 11  Stop Func 9.

*Func 12  D/A Initialization
*Func 13  Perform D/A conversion with software data transfer.

 Func 14  Perform D/A with DMA data transfer.
 Func 15  Get Func 14's operational status.
 Func 16  Stop Func 14.
 Func 17  Perform D/A conversion with interrupt data transfer.

 Func 18  Get Func 17's operational status.
 Func 19  Stop Func 17.

*Func 20  D/I initialization.
*Func 21  Perform digital input with software data transfer.

 Func 22  Perform digital input with DMA data transfer.
 Func 23  Get Func 22's operational status.
 Func 24  Stop Func 22.
 Func 25  Perform digital input with interrupt data transfer.

 Func 26  Get Func 25's operational status.
 Func 27  Stop Func 25.

*Func 28  D/O initialization.
*Func 29  Perform digital output with software data transfer.
*Func 30  Read back current digital output status.
 Func 31  Perform digital output with DMA data transfer.
 Func 32  Get Func 31's operational status.
 Func 33  Stop Func 31.
 Func 34  Perform digital output with interrupt data transfer.
```



```
 Func 35  Get  Func 34's operational status.
 Func 36  Stop Func 34.

 Func 37  Timer initialization.
 Func 38  Timer interrupt enable.
 Func 39  Timer interrupt disable.
 Func 40  Frequency measurement start
 Func 41  Get the Func 40's status.
 Func 42  Stop Func 40.
 Func 43  Event count start.
 Func 44  Read event count.
 Func 45  Stop event count Func 43.
 Func 46  Pulse output start.
 Func 47  Pulse output stop.
 Func 48  One-shot pulse output.
 Func 49  Time interval measurement start.
 Func 50  Get Func 49's status.
 Func 51  Stop Func 49.
*Func 96  Daughter board A/D initialization.
*Func 97  Perform daughter board A/D conversion with software or
          interrupt data transfer.

*Func 98  Get Func 97's status.
*Func 99  Stop Func 97.
*Func 100 Block channel scan initialization.
*Func 101 Perform Block channel scan with software data transfer.
*Func 105 Perform Block channel scan with interrupt data
          transfer.
*Func 106 Get Func 105's status.
*Func 107 Stop Func 105.
```